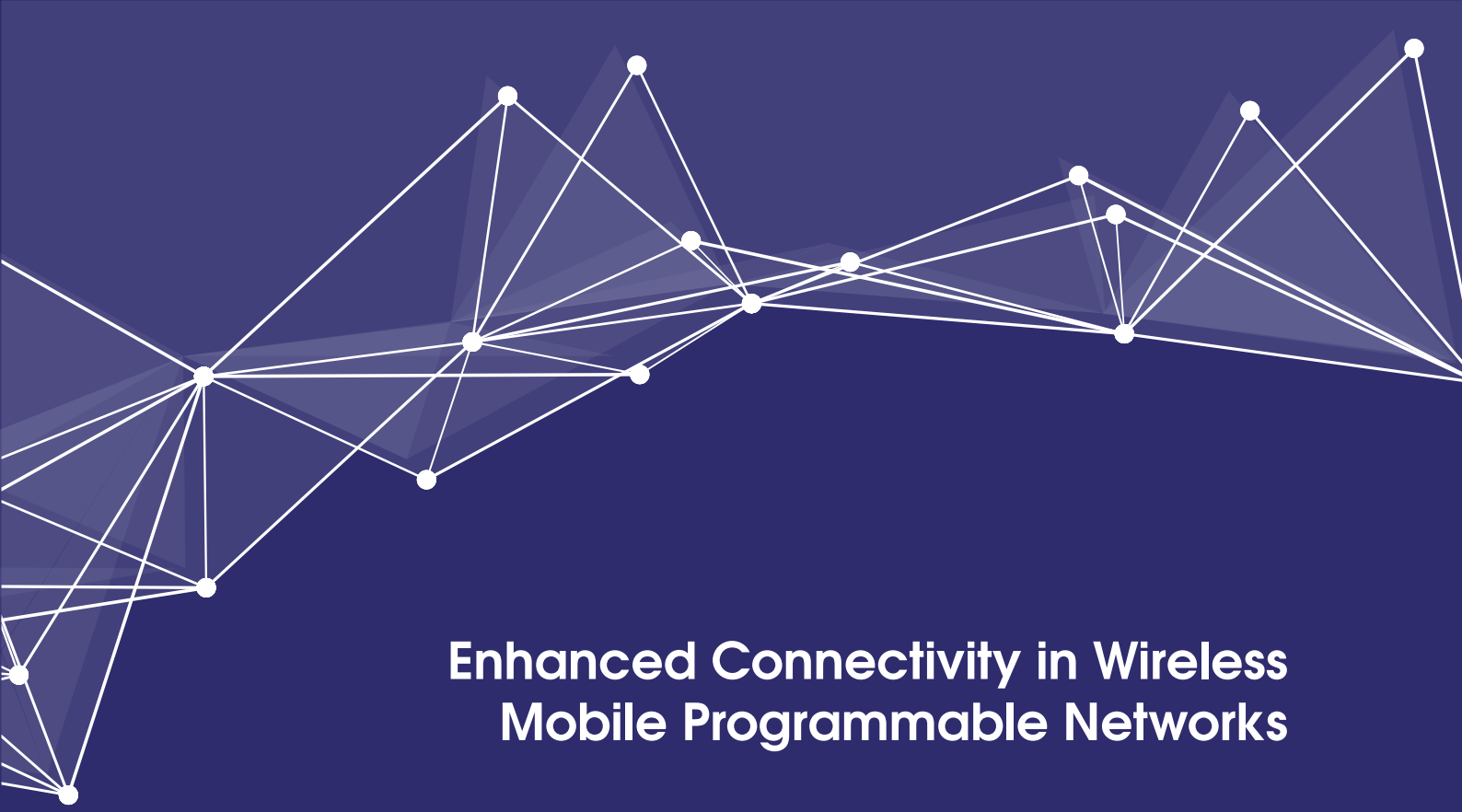


Enhanced Connectivity in Wireless Mobile Programmable Networks

Author: Luca Cominardi

In partial fulfilment of the requirements for the degree of
Doctor in Telematics Engineering

Universidad Carlos III de Madrid

Advisor: Dr. Carlos J. Bernardos

January 2019

Enhanced Connectivity in Wireless Mobile Programmable Networks

Author: Luca Cominardi

In partial fulfilment of the requirements for the degree of
Doctor in Telematics Engineering

Universidad Carlos III de Madrid

Advisor: Dr. Carlos J. Bernardos

January 2019

PUBLISHED BY UNIVERSIDAD CARLOS III DE MADRID

First printing, January 2019

Opportunities are usually disguised as hard work,
so most people don't recognize them
— Ann Landers

Talk is cheap. Show me the code.
— Linus Torvalds

Acknowledgements

The path that led me to the realisation of this thesis has been long and twisted and it spanned across several years and multiple countries with many ups and downs. It started in Madrid, Spain, as a PhD student at the IMDEA Networks Institute. Then, after a couple of years, it took a sharp turn that brought me to London, United Kingdom, at InterDigital Europe Labs in the position of senior research engineer in the field of 5G Radio Access Network (RAN). After the major event that occurred in the United Kingdom on 23 June 2016 (i.e., United Kingdom European Union membership referendum), the path took a second significant detour that conducted me back to Madrid, Spain. This time as a research assistant in the NETCOM research group of the Telematics Engineering department at the University Carlos III of Madrid. During this whole time, regardless of the country where I was living and the company for which I was working, I have been involved in several international projects that materialised in a fruitful collaboration with many people from different countries and continents: Italy, Spain, Germany, United States, Greece, Taiwan, Sweden, Brazil, China – just to name a few.

Among all the great people I have been working with, my exceptional gratitude and sincere appreciation goes to my advisor Carlos J. Bernardos and to Antonio de la Oliva who both flanked me throughout the development of this thesis. Their support and guidance has been fundamental for reaching the many achievements this thesis builds upon. A special thanks also goes to Alain Mourad who guided and advised me during the tumultuous times of my stay in London. I would also like to express my recognition to all my co-authors who have always fostered constructive discussion and distinctive point of views. I am also greatly indebted to my colleagues and friends that made this journey a pleasant one. Especially, I'm very grateful to have shared the path since the very beginning with Iñaki, with whom I discovered to have many more interests in common than networks and computers. Moreover, I would like to also thank Fabio and all my current colleagues and friends at the university who contributed to create a great working environment. Finally, this thesis would not have been possible without the support of my mother, father, and sister who have been visiting me around the world. Last but not least, this thesis is especially dedicated to Letizia, who followed me to Madrid and endured the ups and downs of this thesis on my side.

— Luca Cominardi
Leganés, January 2019

Published and submitted content

In compliance with the principles referring to plagiarism in Law 14/2011 and in the Code of Good Practices of the UC3M Doctoral School, I hereby report a bibliography of articles or other contributions I have (co)authored that are included as part of the thesis and that have been published or submitted for publication.

Published content

J.A. Hernández, D. Larrabeiti, A. de la Oliva, M. Berg, C. Lu, F. Cavaliere, P. Iovanna, Thomas Deiß, **L. Cominardi**, P.-H. Kuo, M.F. Giunta, A. di Giglio, A. Paolicelli, L. Serra, R. Morro, L.A. Neto, P. Chanclou, W. Saint-Aubin, L. Bellot, T. Boukour, J.-C. Plumecoq, J.V. Galán, P. Farkas, V. Jungnickel, J. Baranda, J. Mangues, J. Núñez, M. Miozzo, P. Dini, J.M. Fabrega, M. Svaluto Moreolo, P. Ödling, S. Chang, K.-Y. Lin and R. Chen. *Detailed analysis of the technologies to be integrated in the XFE based on previous internal reports from WP2/3*. Deliverable 2.1. 5G-Crosshaul consortium, June 2016. URL: `http://5g-crosshaul.eu/wp-content/uploads/2018/01/5G-CROSSHAUL_D2.1.pdf` (Retrieved: 10th January 2019)

- This work is partly included and its content is reported in Chapter 2;
- The role of the author of this thesis in the included work relates to the analysis of the Southbound interfaces in the state-of-the-art;
- The material from this source included in this thesis is not singled out with typographic means and references.

L. Cominardi, C. J. Bernardos, P. Serrano, A. Banchs and A. de la Oliva. 'Experimental evaluation of SDN-based service provisioning in mobile networks'. In: *Computer Standards & Interfaces* 58 (2018), pages 158–166. ISSN: 0920-5489. DOI: `10.1016/j.csi.2018.01.004`. URL: `https://www.it.uc3m.es/pablo/papers/pdf/2018-cominardi-csi-evaluation.pdf` (Retrieved: 10th January 2019)

- This work is wholly included and its content is reported in Chapter 1, Chapter 3, Chapter 4, Chapter 5, and Chapter 6;
- The role of the author of this thesis in the included work relates to the design and experimentation of the proposed SDN-framework;
- The material from this source included in this thesis is not singled out with typographic means and references.

L. Cominardi, F. Giust, C. J. Bernardos and A. de la Oliva. 'Distributed mobility management solutions for next mobile network architectures'. In: *Computer Networks* 121 (2017), pages 124–136. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2017.04.008. URL: https://e-archivo.uc3m.es/handle/10016/25830 (Retrieved: 10th January 2019)

- This work is wholly included and its content is reported in Chapter 1, Chapter 3, Chapter 4, Chapter 5, and Chapter 6;
- The role of the author of this thesis in the included work relates to the design and experimentation of the proposed SDN-based Distributed Mobility Management (DMM) solution;
- The material from this source included in this thesis is not singled out with typographic means and references.

L. M. Contreras, **L. Cominardi**, H. Qian and C. J. Bernardos. 'Software-Defined Mobility Management: Architecture Proposal and Future Directions'. In: *Mobile Networks and Applications* 21.2 (April 2016), pages 226–236. ISSN: 1572-8153. DOI: 10.1007/s11036-015-0663-7. URL: https://link.springer.com/article/10.1007/s11036-015-0663-7 (Retrieved: 10th January 2019)

- This work is wholly included and its content is reported in Chapter 1, Chapter 4, Chapter 5, and Chapter 6;
- The role of the author of this thesis in the included work relates to the experimentation and analysis of the SDN controller implementing the proposed DMM solution;
- The material from this source included in this thesis is not singled out with typographic means and references.

T. Deiß, **L. Cominardi**, A. Garcia-Saavedra, P. Iovanna, G. Landi, X. Li, J. Mangues-Bafalluy, J. Núñez-Martínez and A. de la Oliva. 'Packet forwarding for heterogeneous technologies for integrated fronthaul/backhaul'. In: *2016 European Conference on Networks and Communications (EuCNC)*. June 2016, pages 133–137. DOI: 10.1109/EuCNC.2016.7561019. URL: http://5g-crosshaul.eu/wp-content/uploads/2015/05/5G-Crosshaul-WP3-Latest.pdf (Retrieved: 10th January 2019)

- This work is wholly included and its content is reported in Chapter 1, Chapter 7, and Chapter 9;
- The role of the author of this thesis in the included work relates to the analysis and design of the common frame format for packet-based crosshaul traffic;
- The material from this source included in this thesis is not singled out with typographic means and references;

L. Cominardi, J. Baranda, D. Larrabeiti, F. Cavaliere, P. Chanclou, J. Gomes, A. Di Giglio, P. Ödling and H.-W. Chang. '5G-Crosshaul: Towards a unified data-plane for 5G transport networks'. In: *2016 European Conference on Networks and Communications (EuCNC)*. 2016. URL: http://5g-crosshaul.eu/wp-content/uploads/2015/05/WP2-EuCNC16-Reviewers.pdf (Retrieved: 10th January 2019)

- This work is wholly included and its content is reported in Chapter 1, Chapter 7, and Chapter 9;
- The role of the author of this thesis in the included work relates to the analysis of the various transport technologies suitable for an integrated fronthaul and backhaul network;
- The material from this source included in this thesis is not singled out with typographic means and references;

C. J. Bernardos, A. de la Oliva, **L. Cominardi** and L. M. Contreras. *DETNET crosshauling requirements*. Draft draft-bernardos-detnet-crosshaul-requirements-00. Internet Engineering Task Force (IETF), October 2016. URL: https://tools.ietf.org/html/draft-bernardos-detnet-crosshaul-requirements-00 (Retrieved: 10th January 2019)

- This work is wholly included and its content is reported in Chapter 7;
- The role of the author of this thesis in the included work relates to the definition of the requirements for a crosshaul network;
- The material from this source included in this thesis is not singled out with typographic means and references;

L. Cominardi, A. Mourad, D.R. Castor and J.D. Kaewell. *Methods and apparatus for common transport of backhaul and fronthaul traffic*. Patent application WO2017100394A1. June 2017. URL: https://patents.google.com/patent/WO2017100394A1/ (Retrieved: 10th January 2019)

- This work is partly included and its content is reported in Chapter 7;
- The role of the author of this thesis in the included work relates to the design of a common architecture for fronthaul and backhaul networks;
- The material from this source included in this thesis is not singled out with typographic means and references.

L. Cominardi, A. Mourad, R.V. Pragada and D.R. Castor. *Methods, apparatuses and systems directed to common transport of backhaul and fronthaul traffic*. Patent application WO2017147076A1. August 2017. URL: https://patents.google.com/patent/WO2017147076A1/ (Retrieved: 10th January 2019)

- This work is partly included and its content is reported in Chapter 7;
- The role of the author of this thesis in the included work relates to the design of a common forwarding for fronthaul and backhaul networks;
- The material from this source included in this thesis is not singled out with typographic means and references;

L. Cominardi, L. M. Contreras, C. J. Bernardos and I. Berberana. 'Understanding QoS Applicability in 5G Transport Networks'. In: *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. June 2018, pages 1–5. DOI: 10.1109/BMSB.2018.8436847. URL: https://e-archivo.uc3m.es/bitstream/handle/10016/27393/understanding_BMSB_2018_ps.pdf (Retrieved: 10th January 2019)

- This work is wholly included and its content is reported in Chapter 1, Chapter 7, and Chapter 9;
- The role of the author of this thesis in the included work relates to the characterisation and simulation of the traffic requirements in 5G transport networks;
- The material from this source included in this thesis is not singled out with typographic means and references;

M. Perras, **L. Cominardi**, R.G. Gazda and A. Mourad. *Open flow functionality in a software-defined network*. Patent application WO2017142862A1. August 2017. URL: https://patents.google.com/patent/WO2017142862A1/ (Retrieved: 10th January 2019)

- This work is partly included and its content is reported in Chapter 8;
- The role of the author of this thesis in the included work relates to the design of Operations, Administration and Maintenance (OAM) functionality in Open-flow networks;

– The material from this source included in this thesis is not singled out with typographic means and references;

L. Cominardi, O. I. Abdullaziz, K. Antevski, S. B. Chundrigar, R. Gdowski, P. Kuo, A. Mourad, L. Yen and A. Zabala. 'Opportunities and challenges of Joint Edge and Fog Orchestration'. In: *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. April 2018, pages 344–349. DOI: 10.1109/WCNCW.2018.8369006. URL: https://e-archivo.uc3m.es/bitstream/handle/10016/27036/opportunities_WCNCW_2018_ps.pdf (Retrieved: 10th January 2019)

– This work is wholly included and its content is reported in Chapter 1, Chapter 10, and Chapter 12;
– The role of the author of this thesis in the included work relates to the analysis of a joint edge and fog orchestration system;
– The material from this source included in this thesis is not singled out with typographic means and references;

A. Mourad, P.-H. Kuo, D. Rapone, R. Quasso, C. Lu, D. Cederholm, J. M. Roldán Gil, A. Zabala, **L. Cominardi**, A. de la Oliva, A. Colazzo, G. Parmeggiani, S. Talat, R. Gdowski and S. Duqennoy. *5G-CORAL initial system design, use cases, and requirements*. Deliverable 1.1. 5G-Coral consortium, February 2018. URL: http://5g-coral.eu/wp-content/uploads/2018/04/D1.1_final7760.pdf (Retrieved: 10th January 2019)

– This work is partly included and its content is reported in Chapter 11;
– The role of the author of this thesis in the included work relates to the baseline edge and fog integrated architecture and to the evolution from cloud robotics towards fog-assisted robotics;
– The material from this source included in this thesis is not singled out with typographic means and references;

L. Cominardi, I. Úcar Marqués, G. Rigazzi, C. Turyagyenda, J.M. Roldán Gil, A. Zabala Oribe, R. Gdowski, S. Talat, S. Bilal Chundrigar, K.-L. Huang, I. Osamah, G. Baldoni, L.-H. Yen and H. Baghban. *Initial design of 5G-CORAL orchestration and control system*. Deliverable 3.1. 5G-Coral consortium, June 2018. URL: http://5g-coral.eu/wp-content/uploads/2018/06/D3.19802.pdf (Retrieved: 10th January 2019)

– This work is partly included and its content is reported in Chapter 10 and Chapter 11;
– The role of the author of this thesis in the included work relates to the analysis of ETSI MEC and ETSI NFV frameworks in the context of converged edge and fog computing scenarios;
– The material from this source included in this thesis is not singled out with typographic means and references;

Submitted content

L. Cominardi, S. González, A. de la Oliva and C. J. Bernardos. 'Adaptive Telemetry for Software-Defined Mobile Networks'. In: *Journal of Network and Computer Applications* (2018). Submitted

– This work is wholly included and its content is reported in Chapter 1, Chapter 8, and Chapter 9;
– The role of the author of this thesis in the included work relates to the design and experimentation of telemetry functionalities in SDN-based mobile networks;
– The material from this source included in this thesis is not singled out with typographic means and references;

M. Groshev, **L. Cominardi**, K. Antevski, A. de la Oliva, C. J. Bernardos, A. Mourad and R. Gazda. 'A novel approach for multi-access convergence leveraging Edge and Fog computing: an experimentation of a robotics use case'. In: *IEEE Transactions on Services Computing* (2019). To be submitted

- This work is partly included its content is reported in Chapter 1, Chapter 10, and Chapter 12;
- The role of the author of this thesis in the included work relates to the design of a converged Edge and Fog architecture;
- The material from this source included in this thesis is not singled out with typographic means and references;

Other research merits

This chapter first provides a list of additional publications I have (co)authored which are related to this thesis but not included therein. Next, it provides an overview of the various EU-funded projects I was involved in throughout the lifetime of this thesis as well as my role and participation. Finally, it highlights my participation and the role in the Standards Development Organisations and scientific conferences and congresses.

Related publications and submissions

F. Giust, **L. Cominardi** and C. J. Bernardos. 'Distributed mobility management for future 5G networks: overview and analysis of existing approaches'. In: *IEEE Communications Magazine* 53.1 (January 2015), pages 142–149. ISSN: 0163-6804. DOI: 10.1109/MCOM.2015.7010527

D. Wang, E. Katranaras, A. Quddus, N. Sapountzis, **L. Cominardi**, F. Kuo, P. Rost, C. J. Bernardos and I. Berberana. 'SDN-based joint backhaul and access design for efficient network layer operations'. In: *2015 European Conference on Networks and Communications (EuCNC)*. June 2015, pages 214–218. DOI: 10.1109/EuCNC.2015.7194071

F. Cavaliere, P. Iovanna, J. Mangues-Bafalluy, J. Baranda, J. Núñez-Martínez, K.-Y. Lin, H.-W. Chang, P. Chanclou, P. Farkas, J. Gomes, **L. Cominardi**, A. Mourad, A. de la Oliva, J.A. Hernández, D. Larrabeiti, A. Di Giglio, A. Paolicelli and P. Ödling. 'Towards a unified fronthaul-backhaul data plane for 5G The 5G-Crosshaul project approach'. In: *Computer Standards & Interfaces* 51 (2017), pages 56–62. ISSN: 0920-5489. DOI: 10.1016/j.csi.2016.11.005

X. Li, R. Casellas, G. Landi, A. de la Oliva, X. Costa-Perez, A. Garcia-Saavedra, T. Deiß, **L. Cominardi** and R. Vilalta. '5G-Crosshaul Network Slicing: Enabling Multi-Tenancy in Mobile Transport Networks'. In: *IEEE Communications Magazine* 55.8 (2017), pages 128–137. ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1600921

T. Deiß, J. Baranda, **L. Cominardi**, L. M. Contreras, J. Gomes, S. González, P. Iovanna, J. Mangues-Bafalluy, N. Molner, J. Núñez-Martínez, A. de la Oliva and S. Stracca. 'Dataplane Measurements on a Fronthaul (FH) and Backhaul (BH) integrated network'. In: *5th International Workshop on Cloud Technologies and Energy Efficiency in Mobile Communication Networks*

(CLEEN 2017). June 2017. URL: http://eprints.networks.imdea.org/1612/ (Retrieved: 10th January 2019)

A. Reznik, R. Arora, M. Cannon, **L. Cominardi**, W. Featherstone, R. Frazao, F. Giust, S. Kekki, A. Li, D. Sabella, C. Turyagyenda and Z. Zheng. *Developing Software for Multi-Access Edge Computing*. White Paper 20. European Telecommunications Standards Institute (ETSI), September 2017

M. Perras, **L. Cominardi**, R.G. Gazda and A. Mourad. *Mitigating crc calculations in networks that utilize segment routing*. Patent application WO2017172681A1. October 2017. URL: https://patents.google.com/patent/WO2017172681A1/ (Retrieved: 10th January 2019)

D. Rapone, R. Quasso, S. B. Chundrigar, S. T. Talat, **L. Cominardi**, A. De la Oliva, P. Kuo, A. Mourad, A. Colazzo, G. Parmeggiani, A. Z. Orive, C. Lu and C. Li. 'An Integrated, Virtualized Joint Edge and Fog Computing System with Multi-RAT Convergence'. In: *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. June 2018, pages 1–5. DOI: 10.1109/BMSB.2018.8436927

J. Kim, L. M. Contreras, P. Greto, H. Magnusson, H. Woesner, D. Fritzsche, **L. Cominardi** and C. J. Bernardos. 'GiLAN Roaming: Roam Like at Home in a Multi-Provider NFV Environment'. In: *IEEE International Symposium on Networks, Computers and Communications (ISNCC)*. June 2018

K. Antevski, M. Groshev, **L. Cominardi**, C.J. Bernardos, A. Mourad and R. Gazda. 'Enhancing Edge robotics through the use of context information'. In: *Workshop on Experimentation and Measurements in 5G (EM-5G'18)* (December 2018), pages 1–5. DOI: 10.1145/3286680.3286682

L. Cominardi, A. Mourad and P.-H. Kuo. *Slicing switch resources*. Patent application WO2018106868A1. June 2018. URL: https://patents.google.com/patent/WO2018106868A1/ (Retrieved: 10th January 2019)

J. Martín-Pérez, **L. Cominardi**, C. J. Bernardos, A. de la Oliva and A. Azcorra. 'Modeling MEC deployments for low latency streaming scenarios'. In: *IEEE Transactions on Broadcasting* (2018). Submitted

Participation and role in collaborative international projects

iJOIN

The *iJOIN* project (2012-2015) was an EU-funded project (FP7-ICT-2011-8 grant agreement 317941) which introduced the novel concept RAN-as-a-Service (RANaaS), where RAN functionality is flexibly centralised through an open IT platform based on a cloud infrastructure. iJOIN aimed for a joint design and optimisation of access and backhaul, operation and management algorithms, and architectural elements, integrating small-cells, heterogeneous backhaul, and centralised processing. Additionally to the development of technology candidates across PHY, MAC, and the network layer, iJOIN studied the requirements, constraints, and implications for 3GPP LTE-A mobile networks. Finally, iJOIN designed new network operation and management algorithms in the context of RANaaS following the SDN paradigm.

The role and the activities of the author of this thesis in the project are the following:
- Actively participating in the project since September 2013 to July 2015;
- Deputy leader of WP4 on the research and design of novel network layer management enabled by Software Defined Networking (SDN) in Radio Access and Backhaul networks;
- Research and experimental validation of SDN-based Distributed Mobility Management (DMM) mechanisms;
- Demonstration of SDN-based DMM mechanisms at EuCNC 2014, Bologna, Italy and at the Mobile World Congress 2015 (MWC2015), Barcelona, Spain, March 2015;
- The content included in Chapter 3, Chapter 4, and Chapter 5 is related to the scope of this project.

5G-Crosshaul

The *5G-Crosshaul* project (2015-2017) was an EU-funded project (H2020- ICT-2014-2 grant agreement 671598) aimed at developing a 5G integrated backhaul and fronthaul transport network enabling a flexible and software-defined reconfiguration of all networking elements in a multi-tenant and service-oriented unified management environment. This transport network will flexibly interconnect distributed 5G radio access and core network functions, hosted on in-network cloud nodes, through the implementation of: (*i*) a control infrastructure using a unified, abstract network model for control plane integration; (*ii*) a unified data plane encompassing innovative high-capacity transmission technologies and novel deterministic-latency switch architectures.

The role and the activities of the author of this thesis in the project are the following:
- Actively participating in the project since September 2015 to February 2017;
- Research and design of a converged and SDN-based fronthaul and backhaul network with focus on millimetre wave transport links;
- Demonstration of long-distance millimetre wave transport links capable of multiplexing crosshaul traffic in outdoor scenario in Berlin, Germany, September 2016;
- The content included in Chapter 7 and Chapter 8 is related to the scope of this project.

5G Exchange

The *5G Exchange* (5GEx) project (2015-2018) was an EU-funded project (H2020-ICT-2014-2 grant agreement 671636) with the goal of enabling cross-domain orchestration of services over multiple administrations or over multi-domain single administrations. This would allow end-to-end network and service elements to mix in multi-vendor, heterogeneous technology and resource environments. To that end, 5GEx aimed at enabling collaboration between operators with the view of introducing unification via NFV/SDN compatible multi-domain orchestration based on open source software tools and extensions that can be utilised outside the scope of 5GEx.

The role and the activities of the author of this thesis in the project are the following:
- Actively participating in the project since March 2017 to June 2018;
- Prototyping and demonstration of an NFV-based multi-domain roaming solution at Mobile Virtual Network Operator World Congress 2018 (MVNO 2018), Madrid, Spain, April 2018.

5G-Transformer

The *5G-Transformer* project (2017-2019) is an EU-funded project (H2020-ICT-2016-2 grant agreement 761536) aiming at transforming today's mobile transport network into an SDN/NFV-based Mobile Transport and Computing Platform (MTP), which brings the *network slicing* paradigm into mobile transport networks by provisioning and managing MTP slices tailored to the specific needs of vertical industries. The technical approach of the project is twofold: (*i*) enable vertical industries to meet their service requirements within customised MTP slices, and (*i*) aggregate and federate transport networking and computing fabric, from the edge all the way to the core and cloud, to create and manage MTP slices throughout a federated virtualised infrastructure.

The role and the activities of the author of this thesis in the project are the following:
- Actively participating in the project since July 2017 to date;

– Dissemination into ETSI MEC of project-related contributions on edge computing.

5G-CORAL

The *5G-CORAL* project (2017-2019) is an EU-Taiwan project (H2020-ICT-2016-2 grant agreement 761586) leveraging on the pervasiveness of edge and fog computing in the Radio Access Network (RAN) to create a unique opportunity for access convergence. This is envisioned by the means of an integrated and virtualised networking and computing solution where virtualised functions, context-aware services, and user and third-party applications are blended together to offer enhanced connectivity and better quality of experience. The proposed solution contemplates two major building blocks, namely (*i*) the Edge and Fog computing System (EFS) subsuming all the edge and fog computing substrate offered as a shared hosting environment for virtualised functions, services, and applications; and (*ii*) the Orchestration and Control System (OCS) responsible for managing and controlling the EFS, including its interworking with other (non-EFS) domains.

The role and the activities of the author of this thesis in the project are the following:

– Actively participating in the project since September 2017 to date;
– Leader of WP3 on the research and design of orchestration and control systems for edge and fog computing;
– Research and experimental validation of distributed control systems for fog computing tailored to robotics applications;
– Demonstration of a fog-assisted robotics service for cooperative delivery in a commercial shopping mall in Taipei, Taiwan, November 2018;
– Dissemination into ETSI MEC of project-related contributions on edge computing;
– The content included in Chapter 10 is related to the scope of this project.

Participation and role in Standards Development Organisations

ETSI MEC

The ETSI Mobile Edge Computing (MEC) Industry Specification Group (ISG) created the Work Item MEC024 for identifying the necessary support provided by Multi-access Edge Computing for network slicing and, in addition, how the orchestration of resources and services from multiple administrative domains could facilitate that. The work has the intent to collect and analyse the use cases relating to the deployment of MEC functions in combination with network slicing based on the findings from external SDOs, industry collaborations and regional projects, evaluate the gaps from the defined MEC features and functions, and identify the new requirements. When necessary, this may include identifying new MEC services or interfaces, as well as changes to existing MEC services or interfaces, data models, application rules and application requirements. The work item has the intent to recommend the necessary normative work to close these gaps.

The role and the activities of the author of this thesis in ETSI MEC are the following:

– Rapporteur of the ETSI MEC024 Work Item since September 2017 to date.

Technical Programme Committee and Chairmanship

– Local Chair of the Fourth IFIP Conference on Sustainable Internet and ICT for Sustainability (SustainIT 2015), April 2015;
– TPC member of the IEEE 86th Vehicular Technology Conference (IEEE VTC 2017), September 2017;
– TPC member of the First International Workshop on Experimentation and Measurements in 5G (EM-5G), December 2018;
– TPC member of the IEEE Wireless Communications and Networking Conference (IEEE WCNC 2019), April 2019;
– TPC member of the Seventh International Workshop on Cloud Technologies and Energy Efficiency in Mobile Communication Networks (CLEEN 2019), April 2019.

Abstract

The architecture of current operator infrastructures is being challenged by the non-stop growing demand of data hungry services appearing every day. While currently deployed operator networks have been able to cope with traffic demands so far, the architectures for the 5th generation of mobile networks (5G) are expected to support unprecedented traffic loads while decreasing costs associated with the network deployment and operations. Indeed, the forthcoming set of 5G standards will bring programmability and flexibility to levels never seen before. This has required introducing changes in the architecture of mobile networks, enabling different features such as the split of control and data planes, as required to support rapid programming of heterogeneous data planes. Network *softwarisation* is hence seen as a key enabler to cope with such network evolution, as it permits controlling all networking functions through (re)programming, thus providing higher flexibility to meet heterogeneous requirements while keeping deployment and operational costs low. A great diversity in terms of traffic patterns, multi-tenancy, heterogeneous and stringent traffic requirements is therefore expected in 5G networks.

Software Defined Networking (SDN) and Network Function Virtualisation (NFV) have emerged as a basic tool-set for operators to manage their infrastructure with increased flexibility and reduced costs. As a result, new 5G services can now be envisioned and quickly programmed and provisioned

Resumen

La arquitectura de las infraestructuras actuales de los operadores está siendo desafiada por la demanda creciente e incesante de servicios con un elevado consumo de datos que aparecen todos los días. Mientras que las redes de operadores implementadas actualmente han sido capaces de lidiar con las demandas de tráfico hasta ahora, se espera que las arquitecturas de la quinta generación de redes móviles (5G) soporten cargas de tráfico sin precedentes a la vez que disminuyen los costes asociados a la implementación y operaciones de la red. De hecho, el próximo conjunto de estándares 5G traerá la programabilidad y flexibilidad a niveles nunca antes vistos. Esto ha requerido la introducción de cambios en la arquitectura de las redes móviles, lo que permite diferentes funciones, como la división de los planos de control y de datos, según sea necesario para soportar una programación rápida de planos de datos heterogéneos. La softwarisación de red se considera una herramienta clave para hacer frente a dicha evolución de red, ya que proporciona la capacidad de controlar todas las funciones de red mediante (re)programación, proporcionando así una mayor flexibilidad para cumplir requisitos heterogéneos mientras se mantienen bajos los costes operativos y de implementación. Por lo tanto, se espera una gran diversidad en términos de patrones de tráfico, multi-tenancy, requisitos de tráfico heterogéneos y estrictos en las redes 5G.

Software Defined Networking (SDN) y Network Function Virtualisation (NFV) se han convertido en un conjunto de herramientas básicas para que los operadores administren su infraestructura con mayor flexibilidad y menores costes. Como resulta-

in response to user and market necessities, imposing a paradigm shift in the services design. However, such flexibility requires the 5G transport network to undergo a profound transformation, evolving from a static connectivity substrate into a service-oriented infrastructure capable of accommodating the various 5G services, including Ultra-Reliable and Low Latency Communications (URLLC). Moreover, to achieve the desired flexibility and cost reduction, one promising approach is to leverage virtualisation technologies to dynamically host contents, services, and applications closer to the users so as to offload the core network and reduce the communication delay. This thesis tackles the above challenges which are detailed in the following.

A common characteristic of the 5G services is the ubiquity and the almost permanent connection that is required from the mobile network. This really imposes a challenge in the signalling procedures provided to get track of the users and to guarantee session continuity. The mobility management mechanisms will hence play a central role in the 5G networks because of the always-on connectivity demand. Distributed Mobility Management (DMM) helps going towards this direction, by flattening the network, hence improving its scalability, and enabling local access to the Internet and other communication services, like mobile-edge clouds. Simultaneously, SDN opens up the possibility of running a multitude of intelligent and advanced applications for network optimisation purposes in a centralised network controller. The combination of DMM architectural principles with SDN management appears as a powerful tool for operators to cope with the management and data burden expected in 5G networks.

To meet the future mobile user demand at a reduced cost, operators are also looking at solutions such as C-RAN and different functional splits to decrease the cost of deploying and maintaining cell sites. The increasing stress on mobile radio access performance in a context of declining revenues for operators is hence requiring the evolution of backhaul and fronthaul transport networks, which currently work decoupled. The heterogeneity of the nodes and transmission

do, los nuevos servicios 5G ahora pueden planificarse, programarse y aprovisionarse rápidamente en respuesta a las necesidades de los usuarios y del mercado, imponiendo un cambio de paradigma en el diseño de los servicios. Sin embargo, dicha flexibilidad requiere que la red de transporte 5G experimente una transformación profunda, que evoluciona de un sustrato de conectividad estática a una infraestructura orientada a servicios capaz de acomodar los diversos servicios 5G, incluso Ultra-Reliable and Low Latency Communications (URLLC). Además, para lograr la flexibilidad y la reducción de costes deseadas, un enfoque prometedor es aprovechar las tecnologías de virtualización para alojar dinámicamente los contenidos, servicios y aplicaciones más cerca de los usuarios para descargar la red central y reducir la latencia. Esta tesis aborda los desafíos anteriores que se detallan a continuación.

Una característica común de los servicios 5G es la ubicuidad y la conexión casi permanente que se requiere para la red móvil. Esto impone un desafío en los procedimientos de señalización proporcionados para hacer un seguimiento de los usuarios y garantizar la continuidad de la sesión. Por lo tanto, los mecanismos de gestión de la movilidad desempeñarán un papel central en las redes 5G debido a la demanda de conectividad siempre activa. Distributed Mobility Management (DMM) ayuda a ir en esta dirección, al aplanar la red, lo que mejora su escalabilidad y permite el acceso local a Internet y a otros servicios de comunicaciones, como recursos en "nubes" situadas en el borde de la red móvil. Al mismo tiempo, SDN abre la posibilidad de ejecutar una multitud de aplicaciones inteligentes y avanzadas para optimizar la red en un controlador de red centralizado. La combinación de los principios arquitectónicos DMM con SDN aparece como una poderosa herramienta para que los operadores puedan hacer frente a la carga de administración y datos que se espera en las redes 5G.

Para satisfacer la demanda futura de usuarios móviles a un coste reducido, los operadores también están buscando soluciones tales como C-RAN y diferentes divisiones funcionales para disminuir el coste de implementación y mantenimiento de emplazamientos celulares. El creciente estrés en el rendimiento del acceso a la radio móvil en un contexto de menores ingresos para los operadores requiere, por lo tanto, la evolución de las redes de

technologies inter-connecting the fronthaul and backhaul segments makes the network quite complex, costly and inefficient to manage flexibly and dynamically. Indeed, the use of heterogeneous technologies forces operators to manage two physically separated networks, one for backhaul and one for fronthaul. In order to meet 5G requirements in a cost-effective manner, a unified 5G transport network that unifies the data, control, and management planes is hence required. Such an integrated fronthaul/backhaul transport network, denoted as crosshaul, will hence carry both fronthaul and backhaul traffic operating over heterogeneous data plane technologies, which are software-controlled so as to adapt to the fluctuating capacity demand of the 5G air interfaces.

Moreover, 5G transport networks will need to accommodate a wide spectrum of services on top of the same physical infrastructure. To that end, network slicing is seen as a suitable candidate for providing the necessary Quality of Service (QoS). Traffic differentiation is usually enforced at the border of the network in order to ensure a proper forwarding of the traffic according to its class through the backbone. With network slicing, the traffic may now traverse many slice edges where the traffic policy needs to be enforced, discriminated and ensured, according to the service and tenants needs. However, the very basic nature that makes this efficient management and operation possible in a flexible way – the logical *centralisation* – poses important challenges due to the lack of proper monitoring tools, suited for SDN-based architectures. In order to take timely and right decisions while operating a network, centralised intelligence applications need to be fed with a continuous stream of up-to-date network statistics. However, this is not feasible with current SDN solutions due to scalability and accuracy issues. Therefore, an adaptive telemetry system is required so as to support the diversity of 5G services and their stringent traffic requirements.

The path towards 5G wireless networks also presents a clear trend of carrying out computations close to end users. Indeed, pushing contents, applications, and network functions

transporte de backhaul y fronthaul, que actualmente funcionan disociadas. La heterogeneidad de los nodos y las tecnologías de transmisión que interconectan los segmentos de fronthaul y backhaul hacen que la red sea bastante compleja, costosa e ineficiente para gestionar de manera flexible y dinámica. De hecho, el uso de tecnologías heterogéneas obliga a los operadores a gestionar dos redes separadas físicamente, una para la red de backhaul y otra para el fronthaul. Para cumplir con los requisitos de 5G de manera rentable, se requiere una red de transporte única 5G que unifique los planos de control, datos y de gestión. Dicha red de transporte fronthaul/backhaul integrada, denominada "crosshaul", transportará tráfico de fronthaul y backhaul operando sobre tecnologías heterogéneas de plano de datos, que están controladas por software para adaptarse a la demanda de capacidad fluctuante de las interfaces radio 5G.

Además, las redes de transporte 5G necesitarán acomodar un amplio espectro de servicios sobre la misma infraestructura física y el network slicing se considera un candidato adecuado para proporcionar la calidad de servicio necesario. La diferenciación del tráfico generalmente se aplica en el borde de la red para garantizar un reenvío adecuado del tráfico según su clase a través de la red troncal. Con el network slicing, el tráfico ahora puede atravesar muchos fronteras entre "network slices" donde la política de tráfico debe aplicarse, discriminarse y garantizarse, de acuerdo con las necesidades del servicio y de los usuarios. Sin embargo, el principio básico que hace posible esta gestión y operación eficientes de forma flexible – la centralización *lógica – plantea importantes desafíos debido a la falta de herramientas de supervisión necesarias para las arquitecturas basadas en SDN. Para tomar decisiones oportunas y correctas mientras se opera una red, las aplicaciones de inteligencia centralizada necesitan alimentarse con un flujo continuo de estadísticas de red actualizadas. Sin embargo, esto no es factible con las soluciones SDN actuales debido a problemas de escalabilidad y falta de precisión. Por lo tanto, se requiere un sistema de telemetría adaptable para respaldar la diversidad de los servicios 5G y sus estrictos requisitos de tráfico.*

El camino hacia las redes inalámbricas 5G también presenta una tendencia clara de realizar acciones cerca de los usuarios finales. De hecho, acercar los contenidos, las aplicaciones y las fun-

closer to end users is necessary to cope with the huge data volume and low latency required in future 5G networks. Edge and fog frameworks have emerged recently to address this challenge. Whilst the edge framework was more infrastructure-focused and more mobile operator-oriented, the fog was more pervasive and included any node (stationary or mobile), including terminal devices. By further utilising pervasive computational resources in proximity to users, edge and fog can be merged to construct a computing platform, which can also be used as a common stage for multiple radio access technologies (RATs) to share their information, hence opening a new dimension of multi-RAT integration.

ciones de red a los usuarios finales es necesario para hacer frente al enorme volumen de datos y la baja latencia requerida en las futuras redes 5G. Los paradigmas de "edge" y "fog" han surgido recientemente para abordar este desafío. Mientras que el edge está más centrado en la infraestructura y más orientado al operador móvil, el fog es más ubicuo e incluye cualquier nodo (fijo o móvil), incluidos los dispositivos finales. Al utilizar recursos de computación de propósito general en las proximidades de los usuarios, el edge y el fog pueden combinarse para construir una plataforma de computación, que también se puede utilizar para compartir información entre múltiples tecnologías de acceso radio (RAT) y, por lo tanto, abre una nueva dimensión de la integración multi-RAT.

Table of Contents

List of Figures

List of Tables

Introduction

1. Mobile networks landscape

Packet-based mobile networks have experienced a huge success in the last years, with the number of subscribers and traffic volume constantly growing. Several reports like[1] [CIS16][2] show that the mobile traffic growth will not decelerate, but increase 10-fold instead from 2015 by the end of 2020. The envisioned scenario will not only assume a large increase in data volume, but also a profound diversification of traffic and service demands, leading to a new environment for the telecommunication industry which has been identified as the 5th generation of mobile networks. On the one hand, 5G aims at improving the network infrastructure to meet users' demands while reducing the associated deployment and operational costs for network operators. On the other hand, 5G enables a plethora of new services and business opportunities for solutions providers as documented by the 3rd Generation Partnership Project (3GPP)'s New Services and Markets Technology Enablers (SMARTER) [3GP16a][3]. As a result, the 5G wireless networks are expected to support the needs of an hyper-connected society which is continuously demanding very high data rate access, requiring a wider coverage, and offering an increasing number of almost permanently connected devices. In order to cope with this ever-increasing traffic demands, nowadays network technologies are hence experiencing a shift towards *softwarisation*. The key idea is to bring the flexibility and reduced cost of software development to network deployment.

1.1 Software-defined networking

This idea is materialised by the Software Defined Networking (SDN) paradigm [ONF12][4], which moves the intelligence residing in the network elements to a central controller, which implements the network functionality through software. In traditional approaches, the network's control plane is distributed throughout all network devices, while SDN logically centralises the control plane. This removes the need of complex and costly changes in equipment or firmware updates in order to introduce new characteristics in the network, as only a change in the software running at the controller is required. The communication between the centralised controller entity and the merely

[1] This thesis uses the following citation style: [ref.id] is used to identify the reference of the work being cited. The complete list and the extended description of the cited works can be found in the References section at the end of this document (pp. 197). Additionally, in order to help the reader, the full reference is reported as a footnote the first time a work is cited.

[2] CISCO. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015-2020*. White Paper. June 2016.

[3] 3GPP. *Service requirements for next generation new services and markets*. Technical Specification (TS) 22.261 v1.0.0. 3rd Generation Partnership Project (3GPP), December 2016.

[4] ONF. *Software-Defined Networking: The New Norm for Networks*. White Paper. Open Networking Foundation, March 2012.

traffic forwarding devices could be supported by what is called southbound interface protocols. One of the candidate protocols for it is OpenFlow, which is being standardised by the Open Networking Foundation (ONF). The main advantage of this approach is that operators can benefit from an increased *flexibility* to manage their networks and implement new services. Following this new technology (SDN), the 3GPP's architectural study for next generation mobile systems, published in [3GP16b][5], focuses on enhancing the mobile network's core part, considering the evolution of the network towards a distributed and softwarised ecosystem.

Initial work on SDN focused on wired networks, though its advantages are even more important for mobile wireless networks. Indeed, in mobile networks users may change their location over time, and therefore the flexibility provided by SDN is not only beneficial for optimising the traffic distribution inside the network, but it is also useful to adapt the way traffic is steered after users' movement. In particular, some of the benefits of adopting SDN in mobile networks include the following:

- *Modification of traffic engineering policies*: With SDN, an operator can easily change the traffic engineering policies implemented in the network. This can be useful for many reasons, such as for example: (*i*) to select a new gateway for outgoing traffic, (*ii*) route all traffic through a given firewall, or (*iii*) route certain traffic differently.
- *Online traffic optimisation*: SDN does not only allow to flexibly change the traffic engineering policies but it also allows to execute them a finer granularity in terms of (*i*) the timing involved in taking routing decisions, and (*ii*) the traffic flows affected by such decisions. This allows optimising the way traffic is distributed, adapting it as users move to new locations and load changes.
- *Creation of novel services*: SDN also allows treating packets differently based on the user or the application. This can be used to create novel services; for instance, the gateway providing connectivity to a user can be selected (among a pool of available ones) based on the location privacy preferences/requirements of the user. Mobile network services can thus be deployed on the fly, allowing network operators to quickly adapt their network to new requirements.

Note that the above includes a fairly wide range of services which are enabled by SDN, like the ones identified by 3GPP in [3GP16a].

1.2 Distributed mobility management

One service of paramount importance in mobile networks, which intrinsically differentiates them from fixed networks, is the mobility management of the users. Indeed, mobile users are characterised by the fact that they move within the network and may change point of attachment while moving (e.g., antenna they are connected to). The control and data planes of these users are converged at the same Packet Data Network Gateway (PGW) in today's 4G systems, making the 4G architecture highly centralised and hierarchical. The PGW hence is in charge of terminating the mobility signalling of the users as well as forwarding their traffic to and from the mobile network enforcing any policy functions. By doing so, the gateway acts as mobility anchor following the user movements by simply re-routing the packets over tunnels created with the access router where the user terminal is currently connected. But this simplicity comes with some penalties [Cha+14][6]: the mobility anchor represents a single point of failure, it poses scalability issues overloading the network core, and, in general, it leads to sub-optimal paths between the mobile users, also known as Mobile Node (MN), and their communication peers, also known as Correspondent Node (CN). Flattening the network architecture is regarded as one of the most promising approaches to design the architecture of the next generation system, and the Distributed Mobility Management (DMM) paradigm goes precisely in such direction. Both 3GPP and Internet Engineering Task Force

[5] 3GPP. *Study on Architecture for Next Generation System*. Technical Report (TR) 23.799 v14.0.0. 3rd Generation Partnership Project (3GPP), December 2016.

[6] H. Chan et al. *Requirements for Distributed Mobility Management*. Request for Comments (RFC) 7333. Internet Engineering Task Force (IETF), August 2014.

(IETF), which are the main standardisation bodies in this area, have looked and are still looking at DMM-alike solutions.

As a matter of fact, signalling load is one of the most problematic aspects in existing mobile core networks [Met+14][7], [Net12][8]. On the one hand, the number of mobile devices supported by the network (which is the final responsible for their reachability as they move) is ever increasing. On the other hand, the need to increase both capacity and coverage is fuelling the deployment of smaller cells that in turn produce more signalling because of more frequent handover events, regardless the users are actually generating data traffic or not. The main limitation today resides in the control part and not in the data plane, which has enough throughput capacity for traffic forwarding, then driving to a constant upgrade of the mobile core network elements. Specifically, the Mobility Management (MM) is a task that imposes serious constraints to the processing capacity of the existing mobile core network elements. Even if a mobile terminal is not active in a communication, the network should manage its mobility and provide the network resources for facilitating such communication when it is set up. In this situation it is reasonable to look for decoupling the control plane from the data plane in mobile core networks. In fact, mobile protocols already imply such conceptual separation. However, the common incarnation of the mobile core nodes today is the vertical integration of both control and data plane functions in special-purpose and monolithic hardware. A Mobile Packet Core (MPC) architecture based on SDN has been already proposed in [Sam+15][9] describing the evolution of the existing MPC architecture towards a new architecture with the separation of control and user plane by using a programmable network infrastructure.

1.3 Fronthaul and backhaul

The network infrastructure for mobile networks traditionally envisages a backhaul segment for assuring connectivity between the core network and access network. Such segment is typically composed of several sub-systems including heterogeneous wired/wireless forwarding networks and fibre-based aggregation/routing networks, thus including several packet nodes, such as switches, routers, aggregators, etc. This heterogeneity leads to the use of various transport protocols for transporting packets between these nodes such as carrier-grade Ethernet, Optical Transport Network (OTN), Synchronous Digital Hierarchy (SDH), Multiprotocol Label Switching (MPLS), Internet Protocol (IP), etc. The packets transmitted on the backhaul segment are also referred to as backhaul traffic. Recently, a new network segment called fronthaul has emerged, as the result of more centralised radio access network (C-RAN) architectures where the Evolved Node B (eNB) is split into two elements, a Remote Radio Head (RRH) and a Baseband processing Unit (BBU). The RRH simply keeps the RF functions necessary for the signal radiation at the cell site while the BBU takes all the baseband heavy computational functions to a separate location. The simplicity of the antenna sites would allow a more cost effective and flexible deployments of the 5G Radio Access Network (RAN), which is also expected to be denser so as to increase the spectrum re-utilisation and capillarity. To enable this functional split, new protocol interfaces have been designed, such as Common Public Radio Interface (CPRI) [CPR15][10], enhanced CPRI (eCPRI) [CPR18][11], Open Radio equipment Interface (ORI) [ETS14][12], and Next Generation Fronthaul Interface

[7] F. Metzger et al. 'Exploratory Analysis of a GGSN's PDP Context Signaling Load'. In: *Journal of Computer Networks and Communications* 2014 (2014). DOI: 10.1155/2014/526231.

[8] Nokia Siemens Networks. *Signaling is growing 50% faster than data traffic*. White Paper. 2012.

[9] M. R. Sama et al. 'Software-defined control of the virtualized mobile packet core'. In: *IEEE Communications Magazine* 53.2 (February 2015), pages 107–115. ISSN: 0163-6804. DOI: 10.1109/MCOM.2015.7045398.

[10] CPRI. *CPRI Specification*. Interface Specification v7.0. Common Public Radio Interface (CPRI), October 2015.

[11] CPRI. *Common Public Radio Interface: eCPRI Interface Specification*. Interface Specification v1.2. Common Public Radio Interface (CPRI), June 2018.

[12] ETSI. *Requirements for Open Radio equipment Interface (ORI)*. Group Specification (GS) GS ORI 001 V4.1.1. European Telecommunications Standards Institute (ETSI), October 2014.

(NGFI) [IEE16b][13]. The data generated by these interfaces and then transported on the fronthaul segment is referred to as fronthaul traffic. While the location of the RRH is determined by the physical environment and deployments, the physical location of the BBU is variable (e.g., fully at the edge, in a local cloud, or fully central cloud). This can create situations where fronthaul and backhaul traffic may eventually share the same physical segment of the transport network. Different functional splits, and hence interfaces, impose different requirements (e.g., bandwidth, latency, jitter, BER) on the fronthaul interface that must be guaranteed within the transport network, which comprises heterogeneous transmission technologies.

The expected heterogeneity requires a compromise between a dedicated control for each type of technology and a common abstraction layer so as to provision end-to-end services and monitor their Quality of Service (QoS) transparently to the underlying technologies. With traditional technologies, the plethora of nodes, their control elements, and the embedded technologies make the fronthaul and backhaul networks quite complex, expensive and inefficient to manage flexibly and dynamically. Moreover, the new considered functional splits further blur the borders between fronthaul and backhaul. Indeed, both fronthaul and backhaul traffic will have to be carried across the same transport infrastructure, even along with the traffic of fixed access networks. As a result, a common fronthaul/backhaul transport network is required to integrate the heterogeneous media and transport technologies, such as microwave, mmWave, dark fibre, leased lines, etc. To that end, 5G transport networks need to evolve towards an adaptive, flexible and software-defined architecture for integrating multi-technology fronthaul and backhaul segments. The resulting integrated fronthaul and backhaul architecture, also referred to as crosshaul, aims to enable a flexible and software-defined reconfiguration of all networking elements through unified data and control planes interconnecting distributed 5G radio access and core network functions. To tackle the above requirements from a network protocol perspective, Ethernet is considered a suitable candidate in terms of performance, costs, and management for providing packet-based services in a transport network [Oli+15][14]. As a matter of fact, the IEEE is extending current Ethernet standards to (*i*) support link speeds up to 200 Gbps [IEE17d][15], and to (*ii*) provide enhancements for time-sensitive traffic (i.e., fronthaul) [CPR18] which are of particular relevance for 5G. Consequently, transport networks will need to provide huge bandwidth, traffic differentiation, and tailored QoS.

1.4 5G services and network slices

In addition to the backhaul and fronthaul traffic, 5G transport networks will hence need to accommodate different kind of services with very distinct requirements [ITU15a][16] on top of the same physical infrastructure. Specifically, these services span across a large variety of new use cases which go beyond the natural evolution of voice and data delivery in 4G mobile networks [3GP16c][17], such as multi-access network integration, even across operators and less trusted networks, Internet of Things (IoT), localised real-time control, vehicular communication, etc. All of this poses significant challenges to the 4G monolithic and centralised network architecture, both in terms of flexibility and scalability, making new services hard to introduce and scale. In addition, sharing the physical network assets through multi-tenancy is seen as a viable path for reducing the ever-increasing costs

[13] IEEE. *Standard for Packet-based Fronthaul Transport Networks*. NGFI - Next Generation Fronthaul Interface 1914.1. Institute of Electrical and Electronics Engineers (IEEE), August 2016.

[14] A. D. La Oliva et al. 'Xhaul: toward an integrated fronthaul/backhaul architecture in 5G networks'. In: *IEEE Wireless Communications* 22.5 (October 2015), pages 32–40. ISSN: 1536-1284. DOI: 10.1109/MWC.2015.7306535.

[15] IEEE. *Physical Layers and Management Parameters for 50 Gb/s, 100 Gb/s, and 200 Gb/s Operation*. Standard for Ethernet 802.3cd. Institute of Electrical and Electronics Engineers (IEEE), December 2017.

[16] ITU-R. *Framework and overall objectives of the future development of IMT for 2020 and beyond*. M Series Mobile, Radiodetermination, Amateur and Related Satellite Services M.2083. International Telecommunication Union - Radiocommunication Sector (ITU-R), October 2015.

[17] 3GPP. *Study on New Services and Markets Technology Enablers*. Technical Report (TR) 22.891 v14.2.0. 3rd Generation Partnership Project (3GPP), September 2016.

involved in the deployment and management of future networks [SCS16][18]. The above is key to understand the new QoS capabilities that are required to be supported in 5G transport networks so as to simultaneously fulfil the disparate traffic and tenant requirements.

The 3GPP groups 5G services in three main categories [3GP18i][19], namely enhanced Mobile Broadband (eMBB), Ultra-Reliable and Low Latency Communications (URLLC), and Massive Internet of Things (MIoT). Each of them present different inherent characteristics spanning from ultra-low latency to high bandwidth and high reliability. According to Next Generation Mobile Networks (NGMN) [NGM16][20], these multiple services may be provided by customised network slices, which provide the necessary traffic treatment over the same physical substrate. Traffic differentiation is enforced at the access border of the network in order to ensure a proper forwarding of the traffic according to its class through the backbone, where it is more feasible to have high capacity. While existing networks can be considered as a continuum from the access to the interconnections points forming an end-to-end path, network slicing breaks this situation since now the end-to-end path becomes a composition of segmented paths within different slices that could even pertain to distinct administrative organisations or providers. This means that the end-to-end path traverses now many edges where the traffic should be enforced, discriminated and ensured, according to the service and tenants needs. Thus, transport networks move from a single-edge continuum towards a multiple-edges structure in 5G. Apart from the technical complexity added, cost implications can be expected, since the specialised and more expensive hardware today used only in the border for implementing fine-grained QoS should be generalised throughout the network.

To effectively and timely react to changes in the service needs (e.g., QoS) and/or in the underlying infrastructure, operators need to put in place a set of constantly-active critical routines in their networks. These procedures are traditionally referred to as Operations, Administration and Maintenance (OAM). Specifically, operation activities are undertaken to keep the network and their services up and running. Administration activities involve keeping track of resources in the network and how they are used. Maintenance activities are focused on facilitating repairs and upgrades in addition to corrective and preventive measures to make the managed network run more effectively. In the last decade, considerable effort was devoted to enrich existing transport technologies, such as MPLS by the IETF and Provider Backbone Bridges (PBB) by the Institute of Electrical and Electronics Engineers (IEEE), with a comprehensive set of OAM tools with the ultimate goal of providing a carrier grade packet-based network to operators. This effort eventually yielded the release of two competing standards: MPLS Transport Profile (MPLS-TP) and PBB Traffic Engineering (PBB-TE). However, those protocols do not offer the necessary adaptability required in 5G networks because of the rigid implementation of the OAM functionalities. To overcome such limitations, new levels of programmability and flexibility are hence required, triggering the adoption of architectures based on the separation of control and data planes as well as virtualisation.

1.5 Edge and fog computing

In addition to the separation of control and data planes (i.e., SDN), the telecommunication industry is embracing virtualisation technologies for evolving towards a cloud-based infrastructure (see for instance the solution reported in [ATT16][21]). This trend has led to the creation of the European Telecommunications Standards Institute (ETSI) Network Function Virtualisation (NFV) Industry Specification Group (ISG) who pioneered the idea of bringing virtualisation capabilities into mobile

[18] K. Samdanis, X. Costa-Perez and V. Sciancalepore. 'From network sharing to multi-tenancy: The 5G network slice broker'. In: *IEEE Communications Magazine* 54.7 (July 2016), pages 32–39. ISSN: 0163-6804. DOI: 10.1109/MCOM.2016.7514161.

[19] 3GPP. *Service requirements for next generation new services and markets*. Technical Specification (TS) 22.261 v16.4.0. 3rd Generation Partnership Project (3GPP), June 2018.

[20] NGMN. *Description of Network Slicing Concept*. White Paper v1.0. Next Generation Mobile Networks Alliance, February 2016.

[21] AT&T. *ECOMP (Enhanced Control, Orchestration, Management & Policy) Architecture White Paper*. White Paper. AT&T Inc., September 2016.

operator networks [ETS16b][22] to increase flexibility in service offerings and network management. By decoupling the network functions from the underlying hardware platform, NFV allows operators to dynamically deploy services in response to the needs of the traffic. In addition to NFV, ETSI Mobile Edge Computing (MEC) ISG brings computing capabilities close to the end users to cope with the ever-increasing amount of data (e.g., generated by IoT) and the low latency required by some use cases (e.g., vehicular communication) [ETS16a][23]. NFV and MEC jointly represent a paradigm shift for mobile operator networks, which evolve from a centralised architecture based on monolithic and hardware-integrated functions to a software-based distributed architecture. Such evolution enables a common hosting environment, namely edge computing, at the network edge characterised by low latency and high bandwidth as well as real-time access to radio network information. Network functions and software applications can be hence deployed close to the end users, thus alleviating congestion at the mobile network core and serving efficiently local purposes, such as data aggregation for IoT, localised real-time control, and single aggregation point for multi-access connectivity.

Recently, fog computing gained considerable traction in the industrial community as demonstrated by the newborns OpenFog consortium [Con17][24] and IEEE working group on fog computing and networking architecture framework [IEE18c][25]. Fog computing distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum which also envisions the collaborative usage of a multitude of end user or near-user edge devices to carry out a substantial amount of those tasks. It is noteworthy that non-stationary and volatile devices are also considered in fog computing, for example when apparatus are hosted on moving devices (e.g., car, train, mobile user) or are battery-powered (e.g., IoT). While edge computing focuses on operator networks and related use cases, fog computing focuses more broadly on enterprise use cases, which may not be necessarily related to mobile networks (e.g., smart cities, remote surveillance, etc.). Nonetheless, edge and fog present a significant synergy: they both focus on bringing networking and computing capabilities closer to the user. Nowadays, edge and fog computing are stand-alone domains that require separate deployments eventually contending for the same physical resources (e.g., spectrum). The lack of integration poses numerous challenges to the effective usage of those resources in addition to the cost-effectiveness of having multiple separate physical deployments. This is particularly true in today's environments where most of the end user devices are equipped with multiple independent Radio Access Technology (RAT) (e.g., LTE and Wi-Fi) that may simultaneously operate over the same spectrum (e.g., LTE in unlicensed spectrum). As a consequence, the lack of integration acts also as a limiting factor in the exploitation of such multi-RAT diversity. Therefore, a proper harmonisation of edge and fog systems is potentially seen as an enabler for a new degree of convergence in the access with the goal of amalgamating the multiple RATs co-existing nowadays.

1.6 Thesis overview

The remainder of this thesis is organised as follows. Chapter 2 presents an in-depth description of SDN technology, which is the pillar this thesis builds upon. Part one aims at experimentally validating how SDN concepts can greatly simplify network operation in future 5G operator networks. This simplification is achieved by allowing to very easily create and modify network services and thus customise network operation subject to the operator's requirements. To that end, an SDN

[22] ETSI. *Network Functions Virtualisation (NFV); Management and Orchestration; Report on Architectural Options*. Group Specification (GS) NFV-IFA 009 v1.1.1. European Telecommunications Standards Institute (ETSI), July 2016.

[23] ETSI. *Mobile Edge Computing (MEC); Framework and Reference Architecture*. Group Specification (GS) 003 v1.1.1. European Telecommunications Standards Institute (ETSI), March 2016.

[24] OpenFog Consortium. *OpenFog Reference Architecture for Fog Computing*. OPFRA001.020817. Architecture Working Group, February 2017.

[25] IEEE. *Standards for Adoption of OpenFog Reference Architecture for Fog Computing*. FOG - Fog Computing and Networking Architecture Framework 1934. Institute of Electrical and Electronics Engineers (IEEE), July 2018.

framework for quick service provisioning is proposed in Chapter 3. Then, Chapter 4 first explores the evolution of DMM towards SDN, including the identification of DMM design principles and challenges. Next, it proposes and analyses a novel SDN-based DMM solution. Chapter 5 presents a prototype of the SDN-based framework in a real-life test-bed, where the implementation cost of novel complex services is evaluated through experimentation. Additionally, the feasibility of the proposed SDN-based DMM solution is assessed experimentally and the various components evaluated. Chapter 6 finally provides the conclusions of part one highlighting the main related publications.

Part two addresses the design and challenges of future 5G transport networks. Specifically, Chapter 7 discusses the candidate transport and multiplexing technologies for enabling a unified data-plane capable of multiplexing fronthaul and backhaul traffic. Moreover, it proposes a multi-layer switch architecture, with its corresponding abstraction of the forwarding behaviour, and a common frame format that ultimately realise the integrated transport network, also referred to as crosshaul. Furthermore, it analyses the impact of different QoS policies in the transport network in case of having multiple network slices, namely eMBB, URLLC, and MIoT, dealing with fixed and mobile traffic under the performance requirements defined in [3GP18i]. To that end, a simulation framework, namely SimPype, has been developed to evaluate the different QoS policies (SimPype details are reported in Appendix A). Next, Chapter 8 first analyses the mismatch between current OAM tools and SDN solutions in mobile networks. Next, it proposes a telemetry solution for software-defined mobile networks capable of adapting to the various service requirements expected in 5G. It therefore presents an experimental validation which shows the benefits of the proposed solution at alleviating the load on the control and data planes, improving the reactiveness to network events, and providing a better accuracy for network measurements. Finally, Chapter 9 concludes the second part of this thesis and reports the main related publications.

Part three widens the scope of network design by encompassing NFV in addition to SDN and by integrating edge and fog computing in the overall 5G system. On the one hand, this vision allows to extend 5G services from purely network-oriented services to a application/user-centric services where computing capabilities are crucial to the effective service delivery. On the other hand, this concept facilitates multi-access convergence by leveraging the pervasiveness of virtualisation capabilities available in the edge and fog computing ecosystem. Chapter 10 first analyses the opportunities and challenges to integrate, federate, and jointly orchestrate the edge and fog ecosystems into a unified framework tailored to 5G service provisioning and multi-access convergence. Next, Chapter 11 presents the conceptual and logical architecture of such integrated edge and fog system where network operator's and user's virtual functions and applications coexist in the same service area so as to provide enhanced Quality of Exprerience (QoE). To exemplify this architecture an reference use case, namely fog-assisted robotics, is proposed. Chapter 12 finally presents the conclusions of part three and highlights the main related publications.

Ultimately, Chapter 13 draws the thesis conclusions by pinpointing the main contributions of this work with regards to advancements to the state of art. The main publications included in this thesis, and the ones related to the addressed topics, are reported and highlighted. Chapter 14 conclusively identifies the gaps that this thesis leaves uncovered and delineates future lines of work.

2. Background on software-defined networking

A logical view of the Software Defined Networking (SDN) architecture is shown in Figure 2.1. The intelligence and control of SDN switches is centralised in SDN controllers. By doing so, an SDN controller has the global view of the network and is capable of controlling, in a vendor-independent way, the network devices, namely SDN switches. These network devices are no longer required to implement and understand many different network protocol standards; instead, they can provide such functionality by accepting instructions from SDN controllers through the *southbound interface*. This yields a significant saving in time and resources, as the network behaviour can be easily controlled by programming it in the centralised controllers rather than using custom configurations in many different devices scattered across the network. The SDN controller is responsible for the maintenance of an abstract resource model of the underlying network which is then exposed to applications via the *northbound interface*, which is commonly implemented through Rest APIs[1]. Applications define the network behaviour and may belong either to the network operator or to clients, the former usually having a broader scope and higher privileges than the latter.

The following as well as the rest of this thesis focuses on the Southbound Interface (SBI), which provides communication and management between network's SDN controller and physical or virtual resources. There are two distinct types of southbound protocols depending on their purpose. The *Control* protocols primarily control the forwarding/routing, which is the core functionality of the switches and routers in the network. The *Management* protocols convey information regarding the configuration and administration of the elements. In a complex SDN network both control and management protocols are present. In some cases, there are control protocols that have their partner management protocol but they are still distinct and decoupled, permitting a higher degree of flexibility. Moreover, an SBI provides an abstraction of the switch's hardware towards the network controller to enable direct expression of network behaviour and requirements. Multiple abstraction models can be adopted depending on the involved functionality, e.g., a forwarding model defines the packet processing semantic on the switch, while a node model defines the device and peripherals abstraction. The following reports the main southbound protocols for controlling and managing the network elements (e.g., switches) and the main abstraction as defined in the main standardisation bodies, namely Internet Engineering Task Force (IETF) and Open Networking Foundation (ONF).

2.1 Southbound interfaces for controlling the forwarding

This section reports the main southbound interfaces intended to control the forwarding tables on the network elements (e.g., switches).

[1] Note that there is no standard protocol defined for the northbound interface.

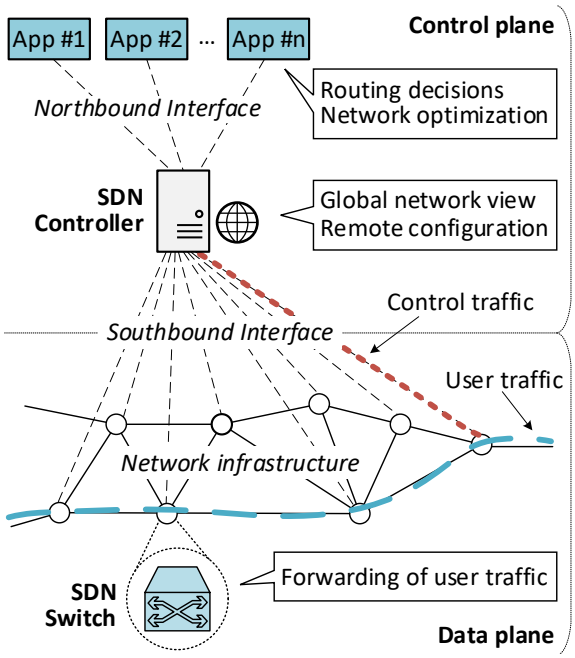


Figure 2.1: Software-defined networking (SDN) architecture.

2.1.1 Forwarding and Control Element Separation Protocol

Forwarding and Control Element Separation (ForCES) defines an architectural framework, and the protocols associated to it, to standardise the information exchange between the control plane and the forwarding plane in a ForCES Network Element (NE). RFC 3654 [KA03][2] defines the ForCES requirements, RFC 3746 [Yan+04][3] defines the ForCES framework, and RFC 5810 [Dor+10][4] specifies the protocol. The ForCES Forwarding Element (FE) is defined by RFC 5812 [HS10][5] and is a logical entity that implements the ForCES protocol and uses the underlying hardware to provide per packet processing and handling as directed by a Control Element (CE). A CE is a logical entity that implements the ForCES Protocol and uses it to instruct one or more FEs on how to process packets. CEs handle functionality such as the execution of control and signalling protocols. Logical Functional Block (LFB) is the basic building block on which the ForCES protocol operates. The LFB is a well-defined, logically separable functional block that resides in an FE and is controlled by the CE via the ForCES protocol. The LFB may reside at the FE's data path and process packets or may be purely an FE control or configuration entity on which the CE operates. Note that the LFB is a functionally accurate abstraction of the FE's processing capabilities, but not a hardware-accurate representation of the FE implementation. The RFC 3654 [KA03] defines requirements that must be satisfied by a ForCES FE model. To summarise, an FE model must define:

- Logically separable and distinct packet forwarding operations in an FE data path (Logical Functional Blocks or LFBs);
- The possible topological relationships (and hence the sequence of packet forwarding operations) between the various LFBs;
- The possible operational capabilities (e.g., capacity limits, constraints, optional features, granularity of configuration) of each type of LFB;

[2] H. Khosravi and T. Anderson. *Requirements for Separation of IP Control and Forwarding*. Request for Comments (RFC) 3654. Internet Engineering Task Force (IETF), September 2003.

[3] L. Yang et al. *Forwarding and Control Element Separation (ForCES) Framework*. Request for Comments (RFC) 3746. Internet Engineering Task Force (IETF), April 2004.

[4] A. Doria et al. *Forwarding and Control Element Separation (ForCES) Protocol Specification*. Request for Comments (RFC) 5810. Internet Engineering Task Force (IETF), March 2010.

[5] J. Halpern and J. Hadi Salim. *Forwarding and Control Element Separation (ForCES) Forwarding Element Model*. Request for Comments (RFC) 5812. Internet Engineering Task Force (IETF), March 2010.

– The possible configurable parameters (e.g., components) of each type of LFB;
– Metadata that may be exchanged between LFBs.

Packets coming into the FE from ingress ports generally flow through one or more LFBs before being transmitted on the egress ports. The result of LFB processing may have an impact on how the packet is to be treated in downstream LFBs. Such differentiated treatment can be conceptualised as having alternative data paths in the FE. For example, the result of a 6-tuple classification performed by a classifier LFB could control which rate meter is applied to the packet by a rate meter LFB in a later stage in the data path. LFB topology is a directed graph representation of the logical data paths within an FE, with the nodes representing the LFB instances and the directed link depicting the packet flow direction from one LFB to the next. In addition to FE model, ForCES defines three execution modes that can be requested for a batch of operations: execute-all-or-none, continue-execute-on-failure, and execute-until-failure. By use of the execute-all-or-none mode, the protocol provides a mechanism for transactional operations within one stand-alone message meeting the ACID requirements [HR83][6].

2.1.2 OpenFlow protocol

OpenFlow is a communications protocol standardised by ONF that gives access to the forwarding plane of a network switch or router over the network. It is the main representative of the Control protocols options for the SBI and it is supported by the main SDN controllers, such as OpenDaylight (ODL)[7], ONOS[8], and Ryu[9]. OpenFlow has influenced the definition of SDN by describing three fundamental paradigms of SDN. The most prominent one is the network architecture with a split user plane and control plane. The second one is a model for rules definition based on packet match and actions. The third one is the OpenFlow protocol itself. The SDN network implements these three paradigms through a central network controller that interacts using OpenFlow with networking devices that implement the match action model. In addition to network controller/switch communication interface, OpenFlow protocol defines the internal architecture of an OpenFlow-enabled Ethernet packet-based switch. The complete switch architecture is composed of several components; among them the main ones are:

– One or more *flow tables* that contain one or more flow entries;
– A *pipeline* which defines how incoming packets interact with the flow tables.

The number of components, the behaviour of the switch, and the interaction with network controllers may vary depending on the adopted OpenFlow protocol version. The first OpenFlow specification limited the protocol to a very concrete scope. Ever since, each subsequent specification of OpenFlow added additional features to the protocol. Table 2.1 summarises the main features supported by each OpenFlow versions. Match fields and Actions table's rows show clearly how OpenFlow has been extended over time to support a greater number of headers, fields, and network protocols. For instance, IPv6 support was first introduced in OpenFlow 1.2, Provider Backbone Bridges (PBB) [IEE09c][10] support was introduced in OpenFlow 1.3.

2.2 Southbound interfaces for managing the network elements

This section reports on the southbound interfaces dedicated to the device configuration. The parameters that can be configured through these interfaces are not related to the forwarding itself but rather to the device.

[6] T. Haerder and A. Reuter. 'Principles of Transaction-oriented Database Recovery'. In: *ACM Comput. Surv.* 15.4 (December 1983), pages 287–317. ISSN: 0360-0300. DOI: 10.1145/289.291.
[7] OpenDaylight: `https://www.opendaylight.org/`
[8] ONOS: `https://onosproject.org/`
[9] Ryu: `https://osrg.github.io/ryu/`
[10] IEEE. *Provider Backbone Bridges*. Standards for Local and metropolitan area networks 802.1ah. Institute of Electrical and Electronics Engineers (IEEE), March 2009.

Table 2.1: OpenFlow versions comparison

PARAMETER	DESCRIPTION	OpenFlow version support					
		1.0	1.1	1.2	1.3	1.4	1.5
Multiple controllers	A switch could be controlled by multiple network controllers	-	-	Y	Y	Y	Y
Auxiliary connections	A switch could have multiple active connections towards the same network controller	-	-	-	Y	Y	Y
Multiple flow tables	OpenFlow pipeline could have more than one flow table	-	Y	Y	Y	Y	Y
Group table	Contains group entries which affect group of flows	-	Y	Y	Y	Y	Y
Meter table	A meter table consists of meter entries which implement simple QoS operations, such as rate limiting	-	-	-	Y	Y	Y
Config. Table-miss	Describes how to manage a packet with no matching rules	-	-	-	Y	Y	Y
Flow table synch	The content is automatically updated by the switch to reflect changes in the flow table it is synchronised with	-	-	-	-	Y	Y
Bundle message	A bundle is a sequence of modification requests that is applied as a single OpenFlow operation meeting ACID requirements [HR83]	-	-	-	-	Y	Y
Ingr/Egr pipelines	Flow tables can be used for ingress or egress processing	-	-	-	-	-	Y
Match fields	Packet match fields used for table lookups (e.g., Ethernet address, IPv4/IPv6 address)	12	22	35	39	41	44
Actions	Actions executed on the packet (e.g., push/pop VLAN, decrement TTL)	10	22	28	30	30	30
Counters	Counters are maintained per table, per-flow, per-port and per queue (e.g., rx/tx bytes, rx/tx error)	22	27	27	40	40	40

2.2.1 OF-Config protocol

The OF-Config protocol is a management protocol that addresses the following controller-switch components:

- OpenFlow *configuration point*: it is in charge of issuing OF-Config commands;
- OpenFlow *capable switch*: a physical or a virtual switching device containing a number of ports and queues;
- OpenFlow *logical switch*: a subset of the ports and queues of an OpenFlow-capable switch which is seen from the outside as a separate OpenFlow-capable switch.

The OF-Config protocol enables configuration of essential artefacts of an OpenFlow logical switch so that an OpenFlow controller can communicate and control the OpenFlow logical switch via the OpenFlow protocol. An OpenFlow-capable switch is intended to be equivalent to an actual physical or virtual network element (e.g. an Ethernet switch) which is hosting one or more OpenFlow data paths by partitioning a set of OpenFlow related resources such as ports and queues among the hosted OpenFlow data paths. The OF-Config protocol enables dynamic association of the OpenFlow related resources of an OpenFlow-capable switch with specific OpenFlow logical switches which are being hosted on the OpenFlow-capable switch. OF-Config does not specify or report how the partitioning of resources in an OpenFlow-capable switch is achieved. OF-Config assumes that resources such as ports and queues are partitioned amongst multiple OpenFlow logical switches such that each OpenFlow logical switch can assume full control over the resources that are assigned to it.

2.2.2 Open vSwitch Database Management protocol

The Open vSwitch Database Management (OVSDB) is a Management protocol described in IETF RFC 7047 [PD13][11]. Open vSwitch is an open-source software switch designed to be used as a vSwitch (virtual switch) in virtualised server environments. A vSwitch forwards traffic between different Virtual Machine (VM) on the same physical host and also forwards traffic between VMs and the physical network. Open vSwitch is open to programmatic extension and control using OpenFlow and the OVSDB (Open vSwitch Database) management protocol which provides an imperative programmatic access. The OVSDB management interface is used to perform management and configuration operations on the OVS instance. Compared to OpenFlow, OVSDB management operations occur on a relatively long time-scale. Examples of operations that are supported by OVSDB include:

- Creation, modification, and deletion of OpenFlow datapaths (i.e., bridges). Multiple OpenFlow datapaths may be available in a single OVS instance;
- Configuration of the set of controllers to which an OpenFlow datapath should connect;
- Configuration of the set of managers to which the OVSDB server should connect;
- Creation, modification, and deletion of ports on OpenFlow datapaths;
- Creation, modification, and deletion of tunnel interfaces on OpenFlow datapaths;
- Creation, modification, and deletion of queues;
- Configuration of Quality of Service (QoS) policies and attachment of those policies to queues;
- Collection of statistics.

Thus, OVSDB is a protocol focusing on the configuration data stored in the database of the switch. Therefore, it is complementary to OpenFlow which operates instead on the flow information stored in the forwarding tables of the vSwitch.

2.2.3 Simple Network Management Protocol

Simple Network Management Protocol (SNMP) is an "Internet-standard protocol for managing devices on IP networks" and is described in RFC 1157 [Cas+90][12]. SNMP exposes management data in the form of variables on the managed systems, which describe the system configuration. These variables can then be queried and set by managing applications. SNMP itself does not define which information (i.e., variables) a managed system should offer. Rather, SNMP uses an extensible design, where the available information is defined by a Management Information Base (MIB). MIBs describe the structure of the management data of a device subsystem; they use a hierarchical namespace containing one or more Object Identifier (OID). Each OID identifies a variable that can be read or set via SNMP. The MIB hierarchy can be depicted as a tree with a nameless root, the levels of which are assigned by different organisations. The top-level MIB OIDs belong to different standard organisations, while lower-level object IDs are allocated by associated organisations. A managed object is one of any number of specific characteristics of a managed device. Managed objects are made up of one or more object instances (identified by their OIDs), which are essentially variables. Two types of managed objects exist:

- Scalar objects define a single object instance;
- Tabular objects define multiple related object instances that are grouped in MIB tables.

There are several versions of SNMP. SNMP v1 is the initial implementation with the most fundamental operations, including Get, GetNext, Set, and Trap. SNMP v2 is a direct update of SNMP v1. New protocols, such as GetBulk and Inform, are added to handle large amounts of data. SNMP v2c, as a sub-protocol of SNMP v2, can be seen as a lighter version of SNMP v2. SNMP v3 adds security and remote configuration capabilities to the previous versions. SNMP is still the most widely used protocol for network equipment fault management and also widely used for perform-

[11] B. Pfaff and B. Davie. *The Open vSwitch Database Management Protocol*. Request for Comments (RFC) 7047. Internet Engineering Task Force (IETF), December 2013.

[12] J. Case et al. *A Simple Network Management Protocol (SNMP)*. Request for Comments (RFC) 1157. Internet Engineering Task Force (IETF), May 1990.

ance management and configuration management. SNMP has demonstrated some advantages over time but it is also showing limitations. Particularly for performance and configuration management it may not provide the same level of flexibility and capabilities as more modern protocols like NETCONF.

2.2.4 Network Configuration Protocol

The Network Configuration Protocol (NETCONF) is a network management protocol developed and standardised by the IETF. It was developed in the NETCONF working group and published as RFC 4741 [Enn06][13] and later revised in RFC 6241 [Enn+11][14] and RFC 7803 [Lei16][15]. NETCONF provides mechanisms to install, manipulate and delete the configuration of network devices and it operates on top of a simple Remote Procedure Call (RPC) layer. The NETCONF protocol can be conceptually partitioned into four layers:

- The Content layer consists of configuration data and notification data;
- The Operations layer defines a set of base protocol operations to retrieve and edit the configuration data;
- The Messages layer provides a mechanism for encoding RPCs and notifications.
- The Secure Transport layer provides a secure and reliable transport of messages between a client and a server.

One particular strength of NETCONF is its support for robust configuration change transactions involving a number of devices. NETCONF has support for a significant part of the networking equipment manufacturers as a substitute of SNMP for configuration management and also for performance management. One of the main advantages of using NETCONF is its support for transactions and, combined with YANG [Bjo10][16], its generic applicability. Understanding SDN as a broader concept than just user plane and control plane separation, NETCONF is very relevant when considering the generic programmability and ability of forwarding devices to be remotely configured and managed.

2.3 Southbound interfaces for interacting with legacy systems

Integrating existing protocols that have matured through industry-wide cooperation and standardisation can help speed up the transition to SDN systems. The following focuses on the analysis of the two main node abstractions most noted in the literature: the Border Gateway Protocol and the Extensible Messaging and Presence Protocol, both from IETF.

2.3.1 Border Gateway Protocol

Border Gateway Protocol (BGP) is a standardised exterior gateway protocol [RLH06][17] designed to exchange routing and reachability information between autonomous systems (AS) on the Internet. The BGP makes routing decisions based on paths, network policies, or rule-sets configured by a network administrator and is involved in making core routing decisions. Currently, some vendors[18] are claiming that the southbound protocol is less important than the operational agility and programmability that SDN architecture aims to offer. These vendors have identified BGP as a potential

[13] R. Enns. *NETCONF Configuration Protocol*. Request for Comments (RFC) 4741. Internet Engineering Task Force (IETF), December 2006.

[14] R. Enns et al. *Network Configuration Protocol (NETCONF)*. Request for Comments (RFC) 6241. Internet Engineering Task Force (IETF), June 2011.

[15] B. Leiba. *Changing the Registration Policy for the NETCONF Capability URNs Registry*. Request for Comments (RFC) 7803. Internet Engineering Task Force (IETF), February 2016.

[16] M. Bjorklund. *YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)*. Request for Comments (RFC) 6020. Internet Engineering Task Force (IETF), October 2010.

[17] Y. Rekhter, T. Li and S. Hares. *A Border Gateway Protocol 4 (BGP-4)*. Request for Comments (RFC) 4271. Internet Engineering Task Force (IETF), January 2006.

[18] `https://searchsdn.techtarget.com/feature/Border-Gateway-Protocol-as-a-hybrid-SDN-protocol`

SDN protocol that can enable network programmability. The controller uses BGP as a Control plane protocol and leverages NETCONF as a Management plane protocol to interact with physical routers, switches and networking services like firewalls. This approach enables SDN to exist in a multi-vendor environment without requiring infrastructure upgrades. The controller operates on multiple levels of abstraction, from routing and bridging topologies to flow-based services. BGP does not include program flows, but operates at a higher level of state like physical and virtual topologies (L2 and L3), security policies, etc. BGP for SDN can offer capital expense savings by allowing network operators to seamlessly integrate existing networks and deployed infrastructure components. Also, the reuse of existing protocols prevents the need for lower-performance software gateways to bridge the physical and virtual worlds, thus reducing network complexity and integrating SDN systems with their existing business logic and processes built around years of experience with BGP.

2.3.2 Extensible Messaging and Presence Protocol

Extensible Messaging and Presence Protocol (XMPP) is a communications protocol for message-oriented middleware based on Extensible Markup Language (XML) and defined in RFC 6120 [Sai11][19]. It enables the near-real-time exchange of structured yet extensible data between any two or more network entities (like SDN controller and switches/routers). XMPP provides a general framework for messaging across a network and was originally developed for instant messaging and on-line presence detection. Not surprisingly, it has a multitude of applications beyond traditional Instant Messaging (IM) and the distribution of Presence data. Indeed, XMPP is emerging as an alternative software-defined networking (SDN) protocol. XMPP can be used by the controller to distribute both control plane and management plane information to the server endpoints and to manage information at all levels of abstraction down to the flow. Traditional protocols are considered necessary for interoperability with legacy networks and systems.

2.4 Southbound interfaces models for node abstraction

In the last years, there have been several initiatives to abstract the forwarding behaviour, abstract the node concept and model them in such a way that it enables programmability of the network. The following focuses on the analysis of the two main node abstractions in literature: the IETF approach to SDN, ForCES; and the standard SDN approach pushed by ONF, OpenFlow.

2.4.1 Forwarding and Control Element Separation

The ForCES model includes capability and state abstraction, the FE and LFB model construction and the unique addressing of the different model structures. The FE/LFB capability model describes the capabilities and capacities of an FE/LFB by specifying the variation in functions supported and any limitations. The state model describes the current state of the FE/LFB, that is, the instantaneous values or operational behaviour of the FE/LFB. The ForCES model includes the constructions for defining the class of LFBs that an FE may support. The definition of such a class provides the information content for monitoring and controlling instances of the LFB class for ForCES purposes. Each LFB model class formally defines the operational LFB components, LFB capabilities and LFB events. The FE model also provides the construction necessary to monitor and control the FE as a whole. For consistency of operation and simplicity, this information is represented as an LFB.

The FE Object LFB class defines the required components to provide coarse grain information at the FE level, i.e., not all possible capabilities or all details about the capabilities of the FE. Part of the FE-level information is the LFB topology, which expresses the logical interconnection between the LFB instances along the data path(s) within the FE. The FE Object also includes information about what LFB classes the FE can support. The ForCES model allows for unique

[19] P. Saint-Andre. *Extensible Messaging and Presence Protocol (XMPP): Core*. Request for Comments (RFC) 6120. Internet Engineering Task Force (IETF), March 2011.

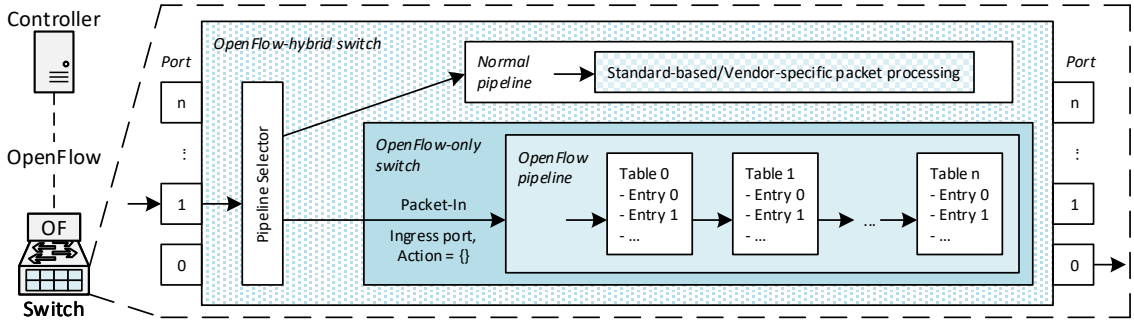


Figure 2.2: OpenFlow switch architecture

identification of the different constructs it defines. This includes identification of the LFB classes, and of LFB instances within those classes, as well as identification of components within those instances. Conceptually, the FE capability model tells the CE which states are allowed on an FE, with capacity information indicating certain quantitative limits or constraints. Thus, the CE has general knowledge about configurations that are applicable to a particular FE. The FE capability model may be used to describe an FE at a coarse level. For example, an FE might be defined as follows:

- FE can handle IPv4 and IPv6 forwarding;
- FE can perform classification based on the following fields: source IP address, destination IP address, source port number, destination port number, etc.;
- FE can perform metering;
- FE can handle up to N queues (capacity); and the FE can add and remove encapsulating headers of types including IPsec, GRE, and L2TP.

While one could try to build an object model to fully represent the FE capabilities, other efforts found this approach to be a significant undertaking. The main difficulty arises in describing detailed limits, such as the maximum number of classifiers, queues, buffer pools, and meters that the FE can provide. A good balance between simplicity and flexibility can be achieved for the FE model by combining coarse-level-capability reporting with an error reporting mechanism. That is, if the CE attempts to instruct the FE to set up some specific behaviour it cannot support, the FE will return an error indicating the problem. Examples of similar approaches include Diffserv [Cha+03][20], which is used for QoS, and a framework for defining policies [Sah+03][21]. The FE state model presents the snapshot view of the FE to the CE. Both LFB capability and state information are defined formally using the LFB modelling XML schema. Capability information at the LFB level is an integral part of the LFB model and provides for powerful semantics. For example, when certain features of an LFB class are optional, the CE needs to be able to determine if those optional features are supported by a given LFB instance. The schema for the definition of LFB classes provides a means for identifying such components.

2.4.2 OpenFlow

OpenFlow uses a completely different approach for node abstraction. An OpenFlow logical switch consists of one or more flow tables and a group table, which perform packet lookups and forwarding, and one or more OpenFlow channels to an external controller. The switch communicates with the controller and the controller manages the switch via the OpenFlow switch protocol. Using the OpenFlow switch protocol, the controller can add, update, and delete flow entries in flow tables, both reactively (in response to packets) and pro-actively. Each flow table in the switch contains a set of flow entries; each flow entry consists of match fields, counters, and a set of instructions

[20] K. Chan et al. *Differentiated Services Quality of Service Policy Information Base*. Request for Comments (RFC) 3317. Internet Engineering Task Force (IETF), March 2003.

[21] R. Sahita et al. *Framework Policy Information Base*. Request for Comments (RFC) 3318. Internet Engineering Task Force (IETF), March 2003.

to apply to matching packets. Flow entries may forward to a port. This is usually a physical port, but it may also be a logical port defined by the switch or a reserved port defined by the OpenFlow specification [ONF15a][22]. Reserved ports may specify generic forwarding actions, such as sending to the controller, flooding, or forwarding using non-OpenFlow methods (e.g., *normal* switch processing). Moreover, switch-defined logical ports may specify link aggregation groups, tunnels or loopback interfaces. OpenFlow ports are the network interfaces for passing packets between OpenFlow processing and the rest of the network.

OpenFlow switches connect logically to each other via their OpenFlow ports, and a packet can be forwarded from one OpenFlow switch to another OpenFlow switch only via an output OpenFlow port on the first switch and an ingress OpenFlow port on the second switch. Hence, the abstraction of a switch promoted by OpenFlow relies on the concept of port. In fact this is one of the main limitations of OpenFlow since for each new technology that is added to OpenFlow, the definition of a new type of port is needed. Currently only Ethernet and Optical ports are defined. It is important to note that this abstraction also has a strong limitation when used for technologies that do not follow the traditional concept of IEEE 802.1 port such as IEEE 802.11. This technology connects a complete IEEE 802.11 network, which may contain multiple access points and stations in any possible configuration, through a portal or integration service which is seen as a single port by switches. This means that the complexity of the topology within the IEEE 802.11 network is completely hidden from OpenFlow, reducing the set of possible actions that can be applied to nodes within the IEEE 802.11 network.

2.5 Southbound interfaces models for forwarding abstraction

Along the same line of the previous section, this section focuses on the analysis of the two main forwarding abstractions most noted in the literature: the IETF approach to SDN, ForCES; and the standard SDN approach pushed by ONF, OpenFlow.

2.5.1 Forwarding and Control Element Separation

ForCES aims to define a framework and associated protocols to standardise information exchange between the control and forwarding plane. Network elements usually expose their functionality to external entities as a single and monolithic instance with a set of defined inputs and expected outputs. In reality, this is not the case and each network element can be dissected in numerous logically separated entities or functionalities that cooperate to provide a given functionality. In the ForCES concept, network elements can be divided into two broad sets of components: (*i*) Control Elements (CEs) in control plane, and (*ii*) Forwarding Elements (FEs) in forwarding plane (or data plane). By defining the communication mechanisms between CEs and FEs, ForCES enables them to be physically separated. This physical separation accrues several benefits to the ForCES architecture. Separate components would allow component vendors to specialise in one component without having to become experts in all components. The ForCES standard protocol also allows the CEs and FEs from different component vendors to interoperate with each other and hence it becomes possible for system vendors to integrate together the CEs and FEs from different component suppliers. This interoperability translates into increased design choices and flexibility for the system vendors. Overall, ForCES enables rapid innovation in both the control and forwarding planes while maintaining interoperability. Scalability is also easily provided by this architecture in that additional forwarding or control capacity can be added to existing network elements without the need for forklift upgrades.

The FE model proposed in ForCES is based on an abstraction using distinct LFBs that are interconnected in a directed graph, and receive, process, modify, and transmit packets along with metadata. The FE and LFB models are designed so that different implementations of the forwarding

[22] ONF. *OpenFlow switch specification: Version 1.5.1*. Technical Specification (TS) 025. Open Networking Foundation, March 2015.

data path can be logically mapped onto the model with the functionality and sequence of operations correctly captured. The LFB topology model for a particular data path implementation must correctly capture the sequence of operations on the packet. The FE model is designed to model the logical processing functions of an FE. The FE model proposed in ForCES includes three components; the modelling of individual LFB (LFB model), the logical interconnection between LFBs (LFB topology), and the FE level attributes, including FE capabilities. The most important block of the ForCES forwarding model is the LFB Class (or type). The LFB is a template that represents a fine-grained, logically separable aspect of FE processing. Most LFBs relate to packet processing in the data path. LFB classes are the basic building blocks of the FE model, which basically interconnects them to obtain a complex behaviour. Complex forwarding functionalities require the CE to be aware of the limitations and capabilities of each NE, since each NE needs to tell the CE which LFBs can be implemented on its hardware. In addition, with all the information of the different LFBs per NE and their desired interconnection, the CE needs to decide which LFBs are implemented on each NE and how to connect everything together so the desired forwarding behaviour is created. This complex procedure may lead to the need of new algorithmic approaches.

2.5.2 OpenFlow

OpenFlow defines two kinds of switches: (*i*) OpenFlow-only which can only process packets following the OpenFlow pipeline, and (*ii*) OpenFlow-hybrid which support OpenFlow and normal switch operations in parallel. These switches should provide an external (i.e., not OF-based) classification mechanism that routes traffic to either the OpenFlow pipeline or the normal pipeline. For example, a switch may use the Virtual LAN (VLAN) tag or input port of the packet to decide whether to process the packet using one pipeline or the other; or it may direct all packets to the OpenFlow pipeline. The OpenFlow pipeline, shown in Figure 2.2, is composed of a set of ingress and egress flow tables, each of them consisting of multiple flow entries. An OpenFlow switch must include at least one ingress flow table. How a packet travels through the OpenFlow pipeline and the set of actions applied to it, while traversing them, defines the packet operation of the OpenFlow switch and the different forwarding behaviours applied to the different flows. The flow tables of an OpenFlow switch are sequentially numbered, starting at 0. Pipeline processing happens in two stages: ingress processing and egress processing. The separation of the two stages is indicated by the first egress table.

Pipeline processing always starts with ingress processing at the first flow table: the packet must be first matched against flow entries of flow table 0. Based on the matching result the packet may be forwarded to an output port or to a different ingress flow table. In case the packet is forwarded to an output port, it may be processed by the first egress flow table assigned to the output port. The use of multiple ingress and egress flow tables allows to implement complex behaviours in a simpler way than having only one single flow table. Each flow table is a collection of flow entries and each flow entry contains the following elements:

- *Match fields*: used to match against packet headers and optionally metadata provided by a previous flow table processing result;
- *Priority*: matching precedence of the flow entry;
- *Counters*: updated when packets are matched;
- *Instructions*: to modify the action set or pipeline processing;
- *Timeouts*: maximum lifetime of the flow entry.

A flow table entry is identified by its match fields and priority: the match fields and priority taken together identify a unique flow entry in a specific flow table. A flow entry instruction may contain actions to be performed on the packet at some point of the pipeline.

2.6 Southbound Interface selection

As it can be evinced from the previous sections, multiple southbound protocols have been designed in literature in order to cover different aspects of the SBI like the forwarding control, network

element management, node abstraction, and forwarding abstraction. Two main standardisation bodies, namely IETF and ONF, have been actively working on the definition of these southbound protocols resulting in ForCES and OpenFlow, respectively. While both protocols present similar characteristics from documentation perspective [Wan+12][23], they differ in one key aspect for what concerns this thesis: the availability of implementations, both open source and hardware-based. Given the experimental approach followed in this document (and related publications), it is therefore important to select an SBI that offers reference implementations suitable for experimentation and validation. To that end, OpenFlow offers many open source implementations and products available on the market, becoming the de-facto standard for the SBI. Unfortunately, this is not true for ForCES. Indeed, no products are available on the market implementing the ForCES protocol at the time of writing of this thesis. The reason can be identified in the different design methodologies and considerations adopted by ForCES and OpenFlow. While the former follows a clean-state design of the control and data planes, the latter departs from existing Ethernet switch hardware architectures. Specifically, OpenFlow exploits the fact that most modern Ethernet switches and routers contain flow-tables (typically built from TCAMs) that run at line-rate to implement applications like firewalls, NAT, QoS, and to collect statistics. While each vendor's flow-table is different, OpenFlow identifies and exploits a common set of functions that run in many switches and routers [McK+08][24]. This design approach allowed the industry and open source community to quickly release OpenFlow products and reference implementations (i.e., Open vSwitch[25]) to the public. For this reason, this thesis focuses on OpenFlow as the main SBI for experimentation and analysis.

[23] Z. Wang et al. *Analysis of Comparisons between OpenFlow and ForCES*. Draft draft-wang-forces-compare-openflow-forces-01. Internet Engineering Task Force (IETF), March 2012.

[24] N. McKeown et al. 'OpenFlow: Enabling Innovation in Campus Networks'. In: *SIGCOMM Comput. Commun. Rev.* 38.2 (March 2008), pages 69–74. ISSN: 0146-4833. DOI: 10.1145/1355734.1355746.

[25] Open vSwitch: `https://www.openvswitch.org/`

Part One

3. A framework for SDN development

This chapter delves into the design of a framework for Software Defined Networking (SDN) development tailored to quick service provisioning. The implementation and evaluation of such framework is then reported in Chapter 5.

3.1 Exemplary use cases for SDN

In order to understand what are the requirements of such framework, a set of representative use cases and services for mobile networks is selected. These use cases cover a wide spectrum of examples, spanning from functionality-oriented to more complex and service-oriented ones. These exemplary use cases will be later used to evaluate the benefits, in terms of service creation time, of properly applying SDN concepts to mobile network architectures. It is worth noticing that the list of identified use cases is not meant to be exhaustive but representative of the SDN context where enhanced flexibility is needed [Agy+14][1].

3.1.1 Location privacy

Tracking the users' location has become very common in recent years as a way to provide customised services. However, this poses several privacy issues [Wer+14][2] which led to the proposal of many counter-measures to preserve the user location [BZO15][3], [KLN15][4]. Notwithstanding, latest distributed mobility management proposals,[5] which envision a flat network architecture with multiple distributed gateways, could unveil more information about the user location than desired. Indeed, such mechanisms route the traffic of a mobile user through the geographically-closest gateway for traffic optimisation reason. One of the security issues raised by such solutions is that they allow tracking the (approximate) location of a mobile user, by monitoring the source IP address of her packets (which reveals the user's gateway), or the service consumed in case it is provided

[1] P. K. Agyapong et al. 'Design considerations for a 5G network architecture'. In: *IEEE Communications Magazine* 52.11 (November 2014), pages 65–75. ISSN: 0163-6804. DOI: 10.1109/MCOM.2014.6957145.

[2] M. Wernke et al. 'A classification of location privacy attacks and approaches'. In: *Personal and Ubiquitous Computing* 18.1 (January 2014), pages 163–175. ISSN: 1617-4917. DOI: 10.1007/s00779-012-0633-z.

[3] C. J. Bernardos, J. C. Zúñiga and P. O'Hanlon. 'Wi-Fi internet connectivity and privacy: Hiding your tracks on the wireless Internet'. In: *2015 IEEE Conference on Standards for Communications and Networking (CSCN)*. October 2015, pages 193–198. DOI: 10.1109/CSCN.2015.7390443.

[4] V. A. Kachore, J. Lakshmi and S. K. Nandy. 'Location Obfuscation for Location Data Privacy'. In: *2015 IEEE World Congress on Services*. August 2015, pages 213–220. DOI: 10.1109/SERVICES.2015.39.

[5] Examples of these approaches are the ones being developed by the IETF Distributed Mobility Management (DMM) WG: https://datatracker.ietf.org/wg/dmm/

close to the gateway [He+17][6]. An obvious way of preserving users' location privacy is to always use the same gateway for a given user, independently of her location; however, this has a very high cost for the operator as traffic would be frequently routed through non-optimal paths.

To address the above problem, one could think of a new privacy service that works as follows: (*i*) for those users that want to preserve their privacy, and contract the corresponding service, a fixed gateway could be used, and (*ii*) for the other users, the best gateway could be used from a traffic engineering perspective. Note that this approach is in line with the recent developments at the Internet Engineering Task Force (IETF), where solutions are being discussed to allow taking into consideration application/user needs when selecting the right anchor/IP address [Yeg+18][7]. Such a solution has a number of advantages: (*i*) it preserves privacy of those users that require it, (*ii*) it has a low cost for the operator in terms of traffic engineering, as efficient routes are used for most traffic, and (*iii*) it provides the means to the operator to receive a revenue from those users willing to contract this service. This is a good example of a new service provided by using modified traffic engineering policies.

3.1.2 Dynamic Service Composition with Network Function Virtualisation

Network Function Virtualisation (NFV) is a new trend that aims at transforming the way telecommunications operators build, manage and exploit their networks, relying on software virtualisation techniques. NFV involves the implementation of network functions in software and its execution on non-specialised and shared hardware. Thus, CAPEX and OPEX are reduced [HIP15][8], as maintenance and updating-related tasks are simplified, and new functions can be introduced via software. SDN is typically seen as complementary with NFV as: (*i*) NFV can support SDN by providing the infrastructure upon which the SDN software can be run, and (*ii*) SDN capacity to create network abstractions can help NFV achieve its goals by enhancing performance. More recently, both the NFV and SDN communities have analysed in more detail how they can co-exist and mutually benefit [ETS15][9], [ONF15b][10]. In an NFV context, SDN is often viewed as a tool to (*i*) enable a flexible and fast interconnection of resources at the NFV Infrastructure (NFVI) level, and (*ii*) facilitate fast configuration of connectivity of a Virtual Network Function (VNF) at service level.

A key feature of NFV is that it enables faster innovation by supporting dynamic, adaptive and quick service deployment. Since network functions can be run on general purpose hardware hosted on data centres, the operator can dynamically react to network and user demand changes by launching services as required and where required. This usually requires the *chaining* of simpler, independent network functions to compose a more complex service. In order to chain these (virtual) network functions, routing needs to be dynamically adjusted in order to forward traffic to the location where the corresponding function is being executed. At this end, SDN can enable a much easier service function chaining/composition by automatically creating the required forwarding paths. Both Open Networking Foundation (ONF) and European Telecommunications Standards Institute (ETSI) NFV have acknowledged this in their latest architecture framework

[6] T. He et al. 'Location Privacy in Mobile Edge Clouds: A Chaff-Based Approach'. In: *IEEE Journal on Selected Areas in Communications* 35.11 (November 2017), pages 2625–2636. ISSN: 0733-8716. DOI: 10.1109/JSAC.2017.2760179.

[7] A. Yegin et al. *On Demand Mobility Management*. Draft draft-ietf-dmm-ondemand-mobility-15. Internet Engineering Task Force (IETF), June 2018.

[8] E. Hernandez-Valencia, S. Izzo and B. Polonsky. 'How will NFV/SDN transform service provider opex?' In: *IEEE Network* 29.3 (May 2015), pages 60–67. ISSN: 0890-8044. DOI: 10.1109/MNET.2015.7113227.

[9] ETSI. *Network Functions Virtualisation (NFV); Ecosystem; Report on SDN Usage in NFV Architectural Framework*. Group Specification (GS) NFV-EVE 005 v1.1.1. European Telecommunications Standards Institute (ETSI), December 2015.

[10] ONF. *Relationship of SDN and NFV: Issue 1*. Technical Report (TR) 518. Open Networking Foundation, October 2015.

updates [ETS15][11], [ONF15b][12].

3.1.3 Multi-tenancy

In order to meet the growing demands of users, network deployments are increasing their density of access points/base stations. At the same time, the cost of deploying and maintaining these new dense deployments is also increasing, reaching a point where operators are looking for innovative ways of reducing their costs. An analysis of how the adoption of NFV and SDN will impact on the reduction of operators' costs is provided in [HIP15], where authors analyse if the intuitive statement often made about the cost reduction originated from the flexibility and simplification enabled by SDN/NFV actually holds. Authors provide a view into the operational costs of a typical service provider and then discuss how the NFV/SDN attributes can be expected to influence the business equation. One of the possible mechanisms identified to reduce costs is the sharing of the network infrastructure, moving into a world where network deployments are multi-tenant by default. In this way, several operators use the resources of a network simultaneously. This requires of complex interactions between the operators and network controllers ensuring the correct utilisation, isolation and sharing of resources.

On-line traffic optimisation mechanisms may be used to ensure the correct isolation and sharing of network resources, so enforcing multi-tenancy in next generation networks. These mechanisms require an up-to-date view of the status of the network. Such requirement can be fulfilled by SDN which enables the continual monitoring of the network. Moreover, SDN allows the abstraction of different virtual network infrastructures, which control can be delegated to each tenant. Each operator sharing the infrastructure can operate and manage their virtual view of the network. This concept has been already explored by the Open Network Foundation in [ONF16][13]. In this document, initial thoughts on how SDN, and OpenFlow in particular, can be used to provide recursion of controllers, enabling the sharing of the infrastructure. Complementary virtualisation technologies (i.e., cloud sharing) are envisioned to support multi-tenancy in NFV along with SDN. Indeed, for a full multi-tenancy environment all the components in the network (switches, data centres, etc.) must support multi-tenancy. Architectures and Proof-of concept demonstrations in the literature, such as [May+16][14], [Vil+15][15], further explore the integration of SDN and NFV in a multi-tenant scenario.

3.1.4 Smart and flexible mobility management

Future mobility management solutions will require increased *flexibility* and shall be capable of adapting to the particular characteristics of the different traffic flows as well as to the heterogeneous nature of the future Radio Access Network (RAN). Indeed, both the IETF and the 3rd Generation Partnership Project (3GPP) are currently working on enhanced mobility architectures and mechanisms providing this additional flexibility, e.g., enabling selective offload of selected traffic to

[11] ETSI. *Network Functions Virtualisation (NFV); Ecosystem; Report on SDN Usage in NFV Architectural Framework*. Group Specification (GS) NFV-EVE 005 v1.1.1. European Telecommunications Standards Institute (ETSI), December 2015.

[12] ONF. *Relationship of SDN and NFV: Issue 1*. Technical Report (TR) 518. Open Networking Foundation, October 2015.

[13] ONF. *SDN architecture: Issue 1.1*. Technical Report (TR) 521. Open Networking Foundation, January 2016.

[14] A. Mayoral et al. 'Multi-tenant 5G Network Slicing Architecture with Dynamic Deployment of Virtualized Tenant Management and Orchestration (MANO) Instances'. In: *ECOC 2016; 42nd European Conference on Optical Communication*. September 2016, pages 1–3.

[15] R. Vilalta et al. 'Network virtualization controller for abstraction and control of OpenFlow-enabled multi-tenant multi-technology transport networks'. In: *2015 Optical Fiber Communications Conference and Exhibition (OFC)*. March 2015, pages 1–3. DOI: 10.1364/OFC.2015.Th3J.6.

local breakout points when possible [3GP11][16], [Che+15][17]. This is a clear example of the need for mechanisms enabling powerful and dynamic traffic engineering policies.

Therefore, the mobility solutions should be capable of (*i*) choosing the right access technology(ies) used by the connected terminals, (*ii*) selecting the gateways and IP addresses assigned to each flow, based on its characteristics and requirements, and, (*iii*) computing the forwarding paths between the radio access points and the used gateways. To achieve this goal, mobility mechanisms require an up-to-date and enriched information regarding the status of the network (e.g., gateway load). SDN is a key technology for gathering, combining, and enriching the information regarding the status of the underlying network.

To that end, [YKS14][18] leverages SDN to offer connectivity management as a service (CMaaS) to application developers and over-the-top service providers to support different types of user mobility at different price levels. On another end, [NBH16][19] introduces an SDN-based mobility management for integrating heterogeneous network technologies and optimising the data transmission costs. Similarly, [Ahm+16][20] increases network flexibility and efficiency by integrating through SDN resource, traffic, and mobility management methods of mobile network services. [Sam+15] summarises the approach developed by the Mobile Packet Core project within the ONF to integrate an SDN architecture into the mobile packet core of an operator, assuming the existence of an NFV context and proposing a unified control architecture considering both SDN and NFV. Finally, 3GPP considers SDN and cloud-based architectures since Release 14 to improve network management [3GP18b][21].

3.2 A functional ONF-based architecture for quick service provisioning

This section first presents the architecture framework defined by the ONF. Then, it proposes an ONF-based architecture of the SDN control plane with the goal of significantly improve flexible and fast service creation in mobile networks.

3.2.1 ONF-based architecture framework

The ONF architecture specifies, at a high level, the reference points and open interfaces that enable the development of software that can control the connectivity provided by a set of network resources and the flow of network traffic though them, along with possible modification of traffic that may be performed in the network. The architecture only describes basic functions that are required, but does not preclude additional functions, allowing a wide range of scenarios and compliant implementations. Consequently, the architecture envisions particular services or applications to interrogate and manipulate the resources in the network [ONF16].

Figure 3.1 illustrates the ONF-based architecture, which is composed of four planes: Data, Controller, Application, and Management, which is transversal to the first three [ONF16]. The *Data* plane comprehends several network resources and is in charge of handling the traffic in the data

[16] 3GPP. *Local IP Access and Selected IP Traffic Offload (LIPA-SIPTO)*. Technical Specification (TS) 23.829 v10.0.1. 3rd Generation Partnership Project (3GPP), October 2011.

[17] G. Chen et al. *Analysis of Failure Cases in IPv6 Roaming Scenarios*. Request for Comments (RFC) 7445. Internet Engineering Task Force (IETF), March 2015.

[18] V. Yazıcı, U. C. Kozat and M. O. Sunay. 'A new control plane for 5G network architecture with a case study on unified handoff, mobility, and routing management'. In: *IEEE Communications Magazine* 52.11 (November 2014), pages 76–85. ISSN: 0163-6804. DOI: 10.1109/MCOM.2014.6957146.

[19] T. T. Nguyen, C. Bonnet and J. Harri. 'SDN-based distributed mobility management for 5G networks'. In: *2016 IEEE Wireless Communications and Networking Conference*. April 2016, pages 1–7. DOI: 10.1109/WCNC.2016.7565106.

[20] I. Ahmad et al. 'New concepts for traffic, resource and mobility management in software-defined mobile networks'. In: *2016 12th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*. January 2016, pages 1–8.

[21] 3GPP. *Architecture enhancements for control and user plane separation of EPC nodes*. Technical Specification (TS) 23.214 v15.3.0. 3rd Generation Partnership Project (3GPP), June 2018.

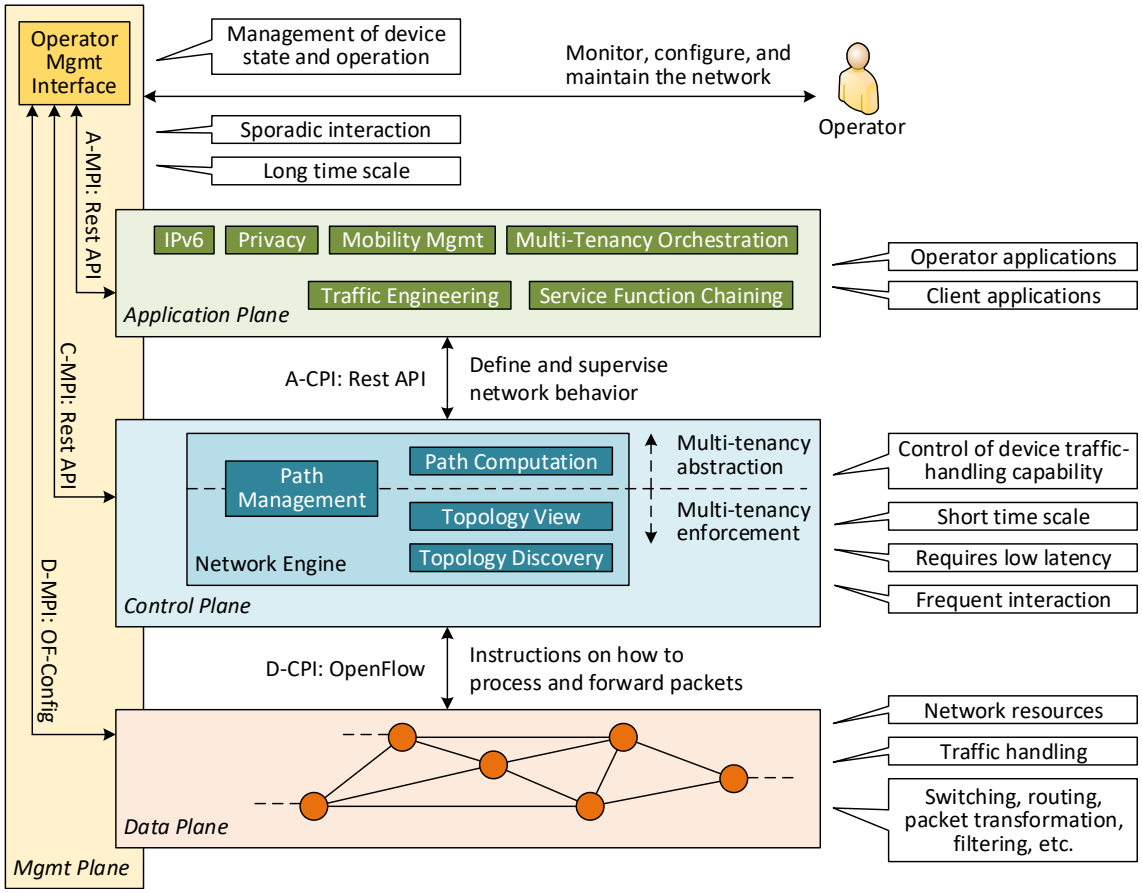


Figure 3.1: Implemented SDN architecture.

path according to the instructions received from the control plane. Examples of operations in this plane are switching, routing, packet encapsulation, etc. Network resources expose their capabilities and receive instructions on how to handle the traffic via the D-CPI interface which connects the Data plane with the control plane. The OpenFlow protocol [ONF15a] is the most widely spread protocol for this interface.

The *Controller* plane is in charge of configuring the appropriate rules on the network resources as to enforce a specific network behaviour. To accomplish this task, the control plane includes several cooperating modules devoted to the creation and maintenance of an abstract resource model of the underlying network which is then exposed to the Application plane via the A-CPI interface. Although there is no standard protocol for the A-CPI interface, it is commonly implemented through Rest Application Programming Interface (API).

The *Application* plane comprises several applications/services whose main goal is to define the network behaviour and may have exclusive control of a set of exposed resources (e.g., network gateway). Applications may belong either to the network operator or to clients with the former usually having a broader scope and higher privileges than the latter. It is worth noticing that applications that primarily support the operation of the data plane (e.g., network topology discovery) are not considered part of the Application plane.

The *Management* plane spans its functionality across all planes and is in charge of monitoring, configuring, and maintaining the network. These functionalities are largely the same as in the control plane, therefore the two planes (Management and Controller) are often seen as a continuum [ONF16]. However, a clear distinction between the two planes resides in the entities interacting with them: a human operator in the Management plane and applications in the control plane. An extensive discussion on the differences between Management and control planes can be

Table 3.1: Control plane implementation details.

MOD	DATA REQUIRED	EVENTS REQUIRED	EVENTS PROVIDED	API
(1)	Connected network nodes	Network node enter/ leave/update; Link appear/disappear/update	Network node enter/ leave/update	Get network node; Get link
(2)	Connected network nodes; Available links	Network node enter/ leave/update; Link appear/disappear/update	Network node enter/ leave/update; Link appear/disappear/update	Set network view; Delete network view; Get network view; Get network node; Get link
(3)	Network view	Network node enter/ leave/update; Link appear/disappear/update	Paths computed	Set traffic class constraint; Set path; Delete path
(4)	Available paths	Paths computed	-	Set path for flow; Delete path for flow

(1) Topology Discovery; (2) Topology View; (3) Path Computation; (4) Path Management;

found at [Hal+15][22]. Subsequently, the two planes differ in terms of (*i*) time-scale and reactiveness, whereas the Management plane works at longer time-scale than the control plane, and (*ii*) scope, with the Management having having greater scope and privilege. Indeed, it is in the scope of the Management plane to perform via the D-MPI interface the initial configuration of the network resources in the Data plane, such as the assignment of the SDN controller(s) they need to connect to and the configuration of queues and ports. This configuration is commonly done via the OF-Config protocol [ONF14][23] in case of OpenFlow-capable switches. In the control plane, the Management uses the C-MPI interface to configure the policies defining the scope of the control given to the SDN applications, to monitor the performance of the system, and to configure the parameters required by the SDN controller modules. In the Application plane, Management configures through the A-MPI the parameters of the applications and the service level agreements. In addition to these interfaces, the Management plane provides a dashboard to network operators for configuring and tuning the network at each layer.

3.2.2 Control plane: design and implementation

ONF provides an architecture framework and some design guidelines, which can then be taken as starting point when specifying a functional and operational system. This section undertakes the challenge of identifying and designing those modules required for a comprehensive, implementable and operating architecture tailored to quick service provisioning. For that reason, a service-oriented architecture is adopted whereas services can be activated by triggers fired upon incoming events. Notably, the principles of service-orientation are independent of any vendor, product, technology, or implementation. In addition, an event-driven communication paradigm is also adopted so as to complement the proposed service-oriented architecture. This paradigm enables the creation of loosely coupled software components and services while increasing responsiveness compared to asynchronous communication, being this aspect fundamental in an environment like a mobile network.

Based on the guidelines set by the general ONF framework, the following reports on the design of the control plane modules marked in blue in Figure 3.1. These modules form the Network Engine, which supports multi-tenancy and implements the following functions: (*i*) discovery of the network topology and building of the resource model for the different tenants, and (*ii*) computation

[22] E. Haleplidis et al. *Software-Defined Networking (SDN): Layers and Architecture Terminology*. Request for Comments (RFC) 7426. Internet Research Task Force (IRTF), January 2015.

[23] ONF. *OpenFlow Management and Configuration Protocol: Version 1.2*. Technical Specification (TS) 016. Open Networking Foundation, 2014.

of the paths within the network and exposure of a management interface for the paths tailored to the different tenants. The Topology Discovery module is in charge of discovering the network topology, while the Topology View builds one or more abstract resource models of the network depending on the multi-tenancy configuration received from the Management plane. The Path Computation module uses these abstract resource models to compute the optimal paths for the different tenants, traffic classes, and requirements, such as latency and bandwidth. The Path Management ensures that each path set-up request received by applications can be accomplished (i.e., Quality of Service (QoS) requirements) and does not violate any constraint (i.e., maximum capacity of the links) or policy (e.g., Service Level Agreement (SLA) of the tenant). Table 3.1 reports the implementation details of each module in terms of data and events required, and the API provided to external modules.

The *Topology Discovery* module implements the Link Layer Discovery Protocol (LLDP) [IEE09d][24] in order to discover the network topology. The Topology Discovery module raises an event whenever a network node enters or leaves the network, or any change occurs on the network node ports. Similarly, an event is raised whenever a link appears or disappears in the network or suffers any changes (i.e., available bandwidth).

The *Topology View* requires the knowledge of the connected network nodes and the available links in order to build an adjacency list graph representation of the network. Such representation is enriched with additional information about network nodes' capabilities (e.g., power profiles, load of the CPU and performance statistics, traffic isolation, etc.). Next, the Topology View module creates an ad-hoc network view for each tenant configured by either the Management plane or the Multi-tenancy application (see Chapter 3.2.3). Such view may be partial and only include a subset of the physical resources or capabilities. With the purpose of maintaining an up-to-date view of the network, the Topology View module subscribes to the events provided by the Topology Discovery module. In turn, the Topology View broadcasts for each tenant (according with their current configuration) an abstracted and enriched version of the events offered by the Topology Discovery. For instance, if a link is not part of a given tenant's view, any event related to that link will not trigger a view update for that tenant. In addition, the module maintains an up-to-date vision of the network by periodically querying the status of each link and network node.

Whenever the Topology View announces a change (or update) in the network for a given tenant, the *Path Computation* module updates the network view for that tenant and (re)computes the paths within the network. The computation occurs for different traffic classes and constraints. MAC bridges [IEE07][25], M2M communications [ETS13a][26], D2D signalling [3GP14][27], and fronthaul traffic [IEE16b] are examples of traffic classes subject to different constraints. Once the module has computed the paths, it raises a path-update event for that tenant. The module is kept as simple as possible implementing a standard Dijkstra's algorithm without any further functionality. If any advanced feature is required, the module's behaviour can be overridden by an external application through the exposed API (i.e., Traffic Engineering).

The *Path Management* module works on the paths provided by Path Computation (which are specific to each tenant and updated periodically) and exposes a Rest interface toward applications. This API is used to request the set-up of paths within the network in the scope of a single tenant. Applications ask Path Management to set-up (or remove) a path for a given flow. A flow may have several requirements such as bandwidth, latency, packet loss and nodes to traverse. At this point, the module ensures that the path can be configured by checking whether the requirements can be

[24] IEEE. *Station and Media Access Control Connectivity Discovery*. Standards for Local and metropolitan area networks 802.1ab-rev. Institute of Electrical and Electronics Engineers (IEEE), January 2009.

[25] IEEE. *IEEE 802.1D - MAC bridges*. IEEE Standards for Local and metropolitan area networks 802.1D. Institute of Electrical and Electronics Engineers (IEEE), August 2007.

[26] ETSI. *Machine-to-Machine communications (M2M); M2M service requirements*. Technical Specification (TS) 102 689 v2.1.1. European Telecommunications Standards Institute (ETSI), July 2013.

[27] 3GPP. *Study on enhancements for infrastructure-based data communication between devices*. Technical Specification (TS) 22.807 v13.0.0. 3rd Generation Partnership Project (3GPP), September 2014.

fulfilled and acknowledges accordingly the requesting application. Besides, the module can be configured by external applications to apply constraints to specific requests. Finally, the module uses a combination of MPLS Transport Profile (MPLS-TP) [Boc+10][28], which is a widely used transport network protocol, and OpenFlow meters to enforce traffic privacy and isolation in the Data plane. It is worth noticing that the choice of using MPLS-TP is mainly driven by the limitation of the OpenFlow switch implementation used for the experimental evaluation in Chapter 5.2. An alternative defined in OpenFlow [ONF15a] is PBB Traffic Engineering (PBB-TE) [IEE09b][29], which however is not supported in the reference OpenFlow switch implementation. More details on how to implement MPLS-TP with OpenFlow switches can be found in [MM18][30]. See [Vai+09][31] for a comparison between MPLS-TP and PBB-TE.

3.2.3 Application Plane: Design and Implementation

The modules implemented in the Application Plane provide the functionality required by the use cases described above in Chapter 3.1. These modules are designed as stand-alone pieces of software which subscribe to the events offered by the control plane and make use of the exposed APIs to trigger changes in the network. These modules are reported as green boxes in Figure 3.1.

The *IPv6* module provides basic IPv6 connectivity to the User Equipment (UE)s. This module is enabled per-tenant basis and is a client application, thus working with limited scope and privileges. The module implements the IPv6 Neighbour Discovery Protocol [Nar+07][32] which is responsible for features such as address auto-configuration, duplicate address detection, and maintaining reachability information. Similarly, an IPv4 module may be implemented providing Dynamic Host Configuration Protocol (DHCP) [Dro97][33] and Address Resolution Protocol (ARP) [Plu82][34] functions.

Mobility Management is responsible of choosing the IP gateways for the UEs in the network. Similarly to IPv6, this module is also a client application and works on per-tenant basis with limited scope and privileges. Whenever a UE connects to the tenant's network (or performs a handover), the module may select different gateways for the UE depending on her profile. For example, one gateway may be selected for real-time traffic while another for best-effort traffic. The selection is also influenced by the proximity of the gateways to the UEs. Doing so, the module offers a two-fold benefit: UEs are always served by optimal gateways and the network core is offloaded. Once the gateways have been selected, Mobility Management asks Path Management to configure the paths for the UE between the selected gateway(s) and the access point(s) the UE is attached to.

The *Privacy* module is a client application and implements the location privacy service. The aim of the service is to maintain the same gateway for those UEs that want to hide their location changes from external nodes. This can be easily achieved by overriding the default gateway selection policy of Mobility Management. Privacy assigns a fixed gateway to the privacy-enabled UEs and communicates the assignment to Mobility Management which will always select that gateway for those UEs.

---

[28] M. Bocci et al. *A Framework for MPLS in Transport Networks*. Request for Comments (RFC) 5921. Internet Engineering Task Force (IETF), July 2010.

[29] IEEE. *Provider Backbone Bridge Traffic Engineering*. IEEE Standards for Local and metropolitan area networks 802.1Q. Institute of Electrical and Electronics Engineers (IEEE), August 2009.

[30] J. Medved and D. Meyer. *MPLS-TP Pseudowire Configuration using OpenFlow 1.3*. Draft draft-medved-pwe3-of-config-01. Internet Engineering Task Force (IETF), June 2018.

[31] R. Vaishampayan et al. 'Application Driven Comparison of T-MPLS/MPLS-TP and PBB-TE - Driver Choices for Carrier Ethernet'. In: *IEEE INFOCOM Workshops 2009* (2009), pages 1–6. DOI: 10.1109/INFCOMW.2009.5072112.

[32] T. Narten et al. *Neighbor Discovery for IP version 6 (IPv6)*. Request for Comments (RFC) 4861. Internet Engineering Task Force (IETF), September 2007.

[33] R. Droms. *Dynamic Host Configuration Protocol*. Request for Comments (RFC) 2131. Internet Engineering Task Force (IETF), March 1997.

[34] David C. Plummer. *An Ethernet Address Resolution Protocol*. Request for Comments (RFC) 826. Internet Engineering Task Force (IETF), November 1982.

Table 3.2: Application plane implementation details.

APP	DATA REQUIRED	EVENTS REQUIRED	EVENTS PROVIDED	API
(1)	Connected switches	Switch enter/leave/ update	Neighbour appear/ disappear	Enable IPv6 Ndisc on switch; Disable IPv6 Ndisc on switch
(2)	Available gateways; Nodes distance; UE profiles	Switch enters/leaves/ update; Paths computed; UE connection	UE anchors selected	Configure gateway selection
(3)	UE profiles	UE profile update	-	-
(4)	Network view	Switch enter/leave/ update; Link appear/ disappear/update	-	Define path; Delete path
(5)	Network view; Paths computed	Switch enters/leaves/ update	-	-
(6)	Network topology	Switch enters/leaves/ update; Link appear/ disappear/update	-	Define network view; Delete network view

(1) IPv6; (2) Mobility Management; (3) Privacy; (4) Traffic Engineering;
(5) Service Function Chaining; (6) Multi Tenancy;

The *Traffic Engineering* module is an operator application and decides how to route different traffic classes within the network. This module works with higher privileges and scopes than client applications and operates either on physical or tenants network. First, the module defines the path for a given traffic class or constraint between two points in the network. Next, the Traffic Engineering module contacts the Path Computation one, and overrides the decision made by the latter. In addition, the network operator can define manually the paths using the module's API. Finally, while many traffic engineering optimisation problems have been proposed in literature [Men+17][35], this module implements the linear programming formulation for Multiprotocol Label Switching (MPLS) networks as proposed in [DMS12][36] to reduce the congestion level of the network.

The *Service Function Chaining* module is an operator application and facilitates the deployment of new services in the network. This module receives the configuration over the Management plane and contacts Path Management for configuring the constraints regarding the requested service. The Service Function Chaining module has access to the up-to-date view of available resources and connectivity from the Topology View module. This view is used to compute a logical overlay connection among the (physical and virtual) network functions composing a given service (chain). This logical overlay is then passed to the Path Management module to compute and implement the required links. The Path Management module does not only consider the logical path imposed by the service needs (e.g., order and location of the network functions that need to be interconnected), but also the requirements on the connectivity itself, e.g., in terms of bandwidth, latency, isolation, geographical/topology constraints, affinity considerations, etc. For example, if a firewall is implemented as virtualised function on a gateway, the Service Function Chaining module interacts with the Path Management module, overriding its default behaviour in such a way that all the UEs traffic associated to that gateway will pass through the firewall location.

The *Multi-tenancy* functionalities of the network are provided by different complementary components. On the one hand, the architecture and internal modules of the controller are multi-

[35] A. Mendiola et al. 'A Survey on the Contributions of Software-Defined Networking to Traffic Engineering'. In: *IEEE Communications Surveys Tutorials* 19.2 (April 2017), pages 918–953. ISSN: 1553-877X. DOI: 10.1109/COMST.2016.2633579.

[36] E. Danna, S. Mandal and A. Singh. 'A practical algorithm for balancing the max-min fairness and throughput objectives in traffic engineering'. In: *2012 Proceedings IEEE INFOCOM*. March 2012, pages 846–854. DOI: 10.1109/INFCOM.2012.6195833.

tenancy oriented since inception. One example of this design, can be found on the Topology View module which, by default, maintains a *real* or physical view of the network, while keeping multiple abstract "views" used by the different tenants. In addition, the allocation of resources to the data plane is done through a combination of an encapsulation supporting resource isolation (e.g., MPLS-TP) and traffic shaping (provided by the OpenFlow meter primitive and software queues configured at the resource level). On the other hand, this functionality is used in combination with the *Multi-tenancy* module. This module is responsible of managing the creation of different virtualised network views and to slice the resources in the network. The multi-tenancy module retrieves the physical network view from the Topology View module and it builds different views of the network according to the configuration received over the Management plane and to the switches capabilities (i.e., traffic isolation, resource reservation, etc.). Each view is a subset of the network resources and the module ensures that the sum of the network views' resources does not exceed the ones of the physical network by implementing a simplified version of the admission control mechanism for new tenant requests as proposed in [Sci+17][37].[38] Once the module assures that the view is consistent, it contacts Topology View and creates the network view for a given tenant. From this moment onwards, all the other modules (i.e., Path Computation and Path Management) will maintain multiple network views, each for tenant. Whenever a tenant contacts one of those modules, the module firstly identifies the corresponding view associated with the tenant, and secondly runs the required procedures as described in the previous section. Similar approaches to this implementation of multi-tenancy in two components, one integrated in the architecture (providing supporting functions) and a second module implemented as an application (providing the bookkeeping of resources) can be found in the literature [Li+17].

[37] V. Sciancalepore et al. 'Mobile traffic forecasting for maximizing 5G network slicing resource utilization'. In: *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. May 2017, pages 1–9. DOI: 10.1109/INFOCOM.2017.8057230.

[38] For the sake of evaluation simplicity, the network capacity constraint is the only constraint considered.

4. An SDN-based Distributed Mobility Management

End-users require maintaining their data connectivity while changing the point of attachment to the network. This feature is enabled by the Mobility Management (MM) protocols that ensure the reachability of the mobile terminal. In-depth surveys of MM protocols can be found in [BR14][1], [AOA12][2]. Mobility Management protocols can be categorised as host-based and network-based MM protocols. In the former case the IP-capable mobile terminals are aware of their IP layer mobility and have to do operations in order to maintain their ongoing communication sessions. In contrast, in the latter case, the functionality to detect and manage the terminal movement resides in the network, and the terminals only perform the standard IP operations. From an operator point of view, this kind of mobility management approach is advantageous because it permits to support mobility without relying on functionality and specific configurations in the mobile node. This thesis focuses on the network-based mobility management protocols. It is worth highlighting that the work presented in the following focuses on Long Term Evolution (LTE) networks since it was performed before the publication of 3rd Generation Partnership Project (3GPP) release 15, which formally defines the 5G system. Nonetheless, the proposed mechanisms are aligned with the ones being currently considered for 5G systems and forthcoming 3GPP release 16 [3GP17][3], [3GP18a][4]. Moreover, the LTE terminology and components for mobility management employed in the following can be mapped 5G systems as described in [3GP17][5], [3GP18j][6].

The mobile operator's core network is in charge of providing IP connectivity and session continuity to the mobile terminal or User Equipment (UE) as moves around. In LTE networks this is provided by the Evolved Packet Core (EPC) [3GP18d][7] while in 5G networks this is provided

[1] R. Bolla and M. Repetto. 'A Comprehensive Tutorial for Mobility Management in Data Networks'. In: *IEEE Communications Surveys Tutorials* 16.2 (February 2014), pages 812–833. ISSN: 1553-877X. DOI: 10.1109/SURV.2013.071913.00140.

[2] I. Al-Surmi, M. Othman and B. Mohd Ali. 'Mobility management for IP-based next generation mobile networks: Review, challenge and perspective'. In: *Journal of Network and Computer Applications* 35.1 (2012), pages 295–315. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2011.09.001.

[3] 3GPP. *5G System - Phase 1; CT WG4 Aspects*. Technical Report (TR) 28.891 v15.0.0. 3rd Generation Partnership Project (3GPP), December 2017.

[4] 3GPP. *5G System; Access and Mobility Management Services; Stage 3*. Technical Specification (TS) 29.518 v15.1.0. 3rd Generation Partnership Project (3GPP), September 2018.

[5] 3GPP. *5G System - Phase 1; CT WG4 Aspects*. Technical Report (TR) 28.891 v15.0.0. 3rd Generation Partnership Project (3GPP), December 2017.

[6] 3GPP. *System Architecture for the 5G System*. Technical Specification (TS) 23.501 v15.3.0. 3rd Generation Partnership Project (3GPP), September 2018.

[7] 3GPP. *General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access*. Technical Specification (TS) 23.401 v14.9.0. 3rd Generation Partnership Project (3GPP), September 2018.

by the 5G core [3GP18j][8]. To provide such IP connectivity, the core network (i.e., EPC or 5G core) extends an overlay tunnel from the Evolved Node B (eNB)/Next Generation NodeB (gNB) in the access to the Packet Data Network Gateway (PGW)/User Plane Function (UPF) deep in the core, to establish connectivity between a mobile terminal to the target external network (e.g., Internet). Those tunnels can be based on different transport protocols along the different EPC/5G core interfaces. In legacy mobile systems, the GPRS Tunnelling Protocol (GTP) [3GP18e][9] has been deployed to provide IP data connectivity from the core network up to the mobile terminal, thus enabling the overlay tunnelling for traversing the core network entities.

The final objective of managing the mobility of the terminal is to guarantee that the IP connectivity is maintained independently of the access point of the user. As a consequence of that, the overlay tunnel has to be re-built to follow the terminal movement as the end user moves. The overlay tunnel which transports the terminal connection from the access to the core is known as bearer. One of the responsibilities of the PGW is to assign each incoming flow to the correct Evolved Packet System (EPS) bearer. To achieve this, each EPS bearer contains a Traffic Flow Template (TFT). Each template contains a set of filters. The filters matches with flows assigned to the bearer. The PGW then looks up the corresponding GTP tunnel, adds certain headers, looks up the IP address of the Serving Gateway (SGW) of user device, and then forwards the packet to the SGW. When the packet arrives at the SGW, the SGW opens the GTP header and reads its Tunnel End-Point Identifier (TEID). Using this information, it identifies the corresponding EPS bearer, and looks up the destination eNB and the TEID. It then establishes another tunnel to forward the packet to the correct eNB.

Another possibility to provide mobility support is the usage of Proxy Mobile IPv6 (PMIPv6). In PMIPv6 [Gun+08][10], mobility support is provided by some specific network entities, namely Mobile Access Gateway (MAG) and Local Mobility Anchor (LMA). The MAG takes care of the mobility signalling on behalf of the Mobile Node (MN)s attached to its links, tracking the mobile nodes as they move, while the LMA stores all the routing information needed to reach the MNs in the PMIPv6 domain by associating each mobile node with the MAG that the MN is using. A tunnel between the LMA and the MAG allows the transfer of traffic from and to the MN. Using PMIPv6, the MN can move across a PMIPv6 domain changing its access link, while keeping its IP address. In EPC architecture the SGW incorporates the MAG functionality and the PGW plays the role of LMA [3GP18g][11].

In centralised mobility management schemes the address of the UE is anchored at home network, so traffic is required to always traverse the central anchor. This indicated that paths between the UEs and its communicating peers would be unnecessarily prolonged. In addition, centralised mobility management imposes scalability challenges because the centralised anchor requires having enough processing capabilities to be able to handle all UEs' traffic. The Internet Engineering Task Force (IETF) is working on mobility management protocols that can benefit from the introduction of Software Defined Networking (SDN) in these environments. Apart from the previously referred PMIPv6 protocol, the Distributed Mobility Management (DMM) group is modifying existing approaches to support a distributed anchoring model. In network-based mobility management, there are two kinds of solutions: one is fully distributed and the other is partially distributed. In the fully distributed approach, each access router acts as both a Mobile Access Gateway (MAG) and a Local Mobility Anchor (LMA). In the partially distributed approach, the data and control planes are separated. But only data plane is distributed. The partially distributed

[8] 3GPP. *System Architecture for the 5G System*. Technical Specification (TS) 23.501 v15.3.0. 3rd Generation Partnership Project (3GPP), September 2018.

[9] 3GPP. *General Packet Radio Service (GPRS); GPRS Tunnelling Protocol (GTP) across the Gn and Gp interface*. Technical Specification (TS) 29.060 v15.2.0. 3rd Generation Partnership Project (3GPP), March 2018.

[10] S. Gundavelli et al. *Proxy Mobile IPv6*. Request for Comments (RFC) 5213. Internet Engineering Task Force (IETF), August 2008.

[11] 3GPP. *Proxy Mobile IPv6 (PMIPv6) based Mobility and Tunnelling protocols; Stage 3*. Technical Specification (TS) 29.275 v15.0.0. 3rd Generation Partnership Project (3GPP), June 2018.

approach is similar to the mobility management scheme used in 3GPP.

Initial proposals have been based on extending existing IP mobility protocols to comply with the requirements of future networks, which include, among others, higher flexibility. To that end, the generic application of the SDN paradigm to wireless and mobile networks has been described in [Ber+14][12], [CH13][13], [LMR12][14]. Specifically, the usage of SDN in this area is expected to simplify the mobility management procedures at both control and data plane. Some of the present challenges of mobile networks in general [CH13][15], and mobility management protocols [AOA12][16] in particular, can be benefited by the logically centralised control approach provided by SDN. By decoupling the control plane from the forwarding element the signalling workload can be offloaded from the forwarding devices as implemented today by consolidating the control in a (logically centralised) single entity. Regarding the data plane, the SDN approach can improve the operation of the network by simplifying the traffic delivery in the network nodes by receiving instructions from the SDN controller, which maintains a complete view of the communication end-to-end, from the terminal to the gateway, and even beyond.

This chapter first presents a Distributed Mobility Management (DMM) solution based on a well known existing IP mobility protocol PMIPv6, which could be referred to as *legacy* solution. This legacy solution, which was first proposed in [GBO14][17] and then contributed to IETF [BOG18][18], is hence used as a baseline to identify the design principles and challenges of a DMM solution embracing the SDN paradigm, which could be in turn considered as *evolutionary* solution. Based on this analysis, a new SDN-based DMM solution is proposed for addressing the identified design principles and specific challenges. A unified methodology, which takes into account protocol specific operations (e.g., number and type of control messages, and mobility model), is proposed to analytically evaluate and compare both DMM solutions in terms of handover cost, scalability, and state space size. The experimental evaluation of the mobility solutions under considerations is then presented in Chapter 5.

4.1 Evolution from IP mobility towards SDN

The main proposition of DMM is simple: distributing mobility anchors by placing multiple ones closer to the location of the user. A considerably large amount of research has been conducted in this area, producing different kinds of solutions. The Distributed Mobility Management Working Group (WG) at the IETF is one of the first and main venues where solutions for distributed IP mobility management are discussed. The group started exploring the DMM problem space by first looking at existing IP mobility protocols, like Mobile IPv6 (MIPv6) [PJA11][19] and PMIPv6 [Gun+08].

[12] C. J. Bernardos et al. 'An architecture for software defined wireless networking'. In: *IEEE Wireless Communications* 21.3 (June 2014), pages 52–61. ISSN: 1536-1284. DOI: 10.1109/MWC.2014.6845049.

[13] C. Chaudet and Y. Haddad. 'Wireless Software Defined Networks: Challenges and opportunities'. In: *2013 IEEE International Conference on Microwaves, Communications, Antennas and Electronic Systems (COMCAS 2013)*. October 2013, pages 1–5. DOI: 10.1109/COMCAS.2013.6685237.

[14] L. E. Li, Z. M. Mao and J. Rexford. 'Toward Software-Defined Cellular Networks'. In: *Proceedings of the 2012 European Workshop on Software Defined Networking*. EWSDN '12. IEEE Computer Society, 2012, pages 7–12. ISBN: 978-0-7695-4870-8. DOI: 10.1109/EWSDN.2012.28.

[15] C. Chaudet and Y. Haddad. 'Wireless Software Defined Networks: Challenges and opportunities'. In: *2013 IEEE International Conference on Microwaves, Communications, Antennas and Electronic Systems (COMCAS 2013)*. October 2013, pages 1–5. DOI: 10.1109/COMCAS.2013.6685237.

[16] I. Al-Surmi, M. Othman and B. Mohd Ali. 'Mobility management for IP-based next generation mobile networks: Review, challenge and perspective'. In: *Journal of Network and Computer Applications* 35.1 (2012), pages 295–315. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2011.09.001.

[17] F. Giust, C. J. Bernardos and A. de la Oliva. 'Analytic Evaluation and Experimental Validation of a Network-Based IPv6 Distributed Mobility Management Solution'. In: *IEEE Transactions on Mobile Computing* 13.11 (November 2014), pages 2484–2497. ISSN: 1536-1233. DOI: 10.1109/TMC.2014.2307304.

[18] C. J. Bernardos, A. de la Oliva and F. Giust. *Proxy Mobile IPv6 extensions for Distributed Mobility Management*. Draft draft-ietf-dmm-pmipv6-dlif-03. Internet Engineering Task Force (IETF), October 2018.

[19] C. Perkins, D. Johnson and J. Arkko. *Mobility Support in IPv6*. Request for Comments (RFC) 6275. Internet Engineering Task Force (IETF), July 2011.

The intention was to investigate possible extensions and adaptations to accepted standard protocols, in order to limit the impact on legacy implementations and equipment. Beyond IETF DMM and the already mentioned 3GPP's activity in [3GP16b], also the Open Networking Foundation (ONF) Wireless and Mobile working group has taken its part, by proposing the adoption of SDN for the design of mobile networks.

As briefly introduced above, most of the first proposals of the DMM WG considered PMIPv6 as the baseline of the solution, as documented in [BOG18], [CP14][20], [SBL14][21]. PMIPv6 is one of the mobility protocols adopted by the 3GPP EPC and provides network-based mobility whose main characteristic is to not require any active participation of the MN to support mobility. It is worth noticing that the MN might be completely unaware of the Layer-3 mobility in place by the network. Other mobility protocols like MIPv6, or its DMM extension proposed in [LBL12][22], provide instead a client-based mobility solution which requires the active involvement of the MN during attachment or handover procedures.

The following section provide a detailed explanation of the PMIPv6-based DMM solution that has been contributed to the IETF [BOG18]. This solution is then taken as basis for comparison since (*i*) it is under standardisation in IETF, (*ii*) it is comparable to many similar solutions which extend existing mobility protocols, and (*iii*) an analytic and experimental evaluation of the solution is available in [GBO14].

4.1.1 IP mobility (PMIPv6) based DMM solution

The key entity in this solution is the DMM Gateway (DMM-GW). The DMM-GW extends the PMIPv6 MAG functions incorporating most of the functionality of the PMIPv6 LMA. Hence, a DMM-GW provides connectivity to IP based services, e.g., Internet, and has the capability of assigning and anchoring IPv6 prefixes. A unique IPv6 prefix pool belongs to each DMM-GW, from which a prefix is assigned to every MN attached to the DMM-GW's access links. In this way, a DMM-GW acts as a plain access router to forward packets to and from the Internet. Moreover, it is provided with mobility anchoring functions, i.e., a DMM-GW is able to maintain the uplink and downlink forwarding for the IP flows that an MN started while attached to that DMM-GW, even after the MN has moved to a new DMM-GW. An external node, referred to as Control Mobility Database (CMD), is used to store the location of the MNs in the domain (i.e., the bindings).

In order to better understand the solution, its main operations are illustrated in Figure 4.1 and are referred to with a number (#). First, a DMM-GW detects an MN attachment by using IPv6 Neighbour Discovery [Nar+07] signalling (1) (typically, an IPv6 host sends a Router Solicitation (RS) message upon joining a link) or by a dedicated link layer detection mechanism. Then, the DMM-GW notifies the CMD about the MN attachment by means of a Proxy Binding Update (PBU) message containing an IPv6 prefix reserved for the MN (2). In case of initial registration, there is no entry available in the CMD's cache for that particular MN, so the CMD registers for the first time the MN, by storing the IPv6 prefix assigned to the MN associated to the MN's location, i.e., the DMM-GW's address. The CMD then acknowledges the operation to the DMM-GW with a Proxy Binding Acknowledgment (PBA) message (3), and the DMM-GW finally delegates the IPv6 prefix to the MN with a Router Advertisement (RA) message (4). Upon a handover, the messages (5,6) are sent, reflecting steps (1,2), but now the CMD receives a new IPv6 prefix in the PBU from the new DMM-GW (6), so the CMD associates the MN with the new prefix and the new location. The old location and prefix are included in a list of *anchoring* DMM-GWs and these parameters are conveyed to the new DMM-GW in the PBA message (7). In parallel, the CMD sends a PBU (8) to

[20] H. Chan and K. Pentikousis. *Enhanced mobility anchoring*. Draft draft-chan-dmm-enhanced-mobility-anchoring-00. Internet Engineering Task Force (IETF), July 2014.

[21] P. Seite, P. Bertin and JH. Lee. *Dynamic Mobility Anchoring*. Draft draft-seite-dmm-dma-07. Internet Engineering Task Force (IETF), August 2014.

[22] J. H. Lee, J. M. Bonnin and X. Lagrange. 'Host-based distributed mobility management support protocol for IPv6 mobile networks'. In: *2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* (October 2012), pages 61–68. ISSN: 2160-4886. DOI: 10.1109/WiMOB.2012.6379140.

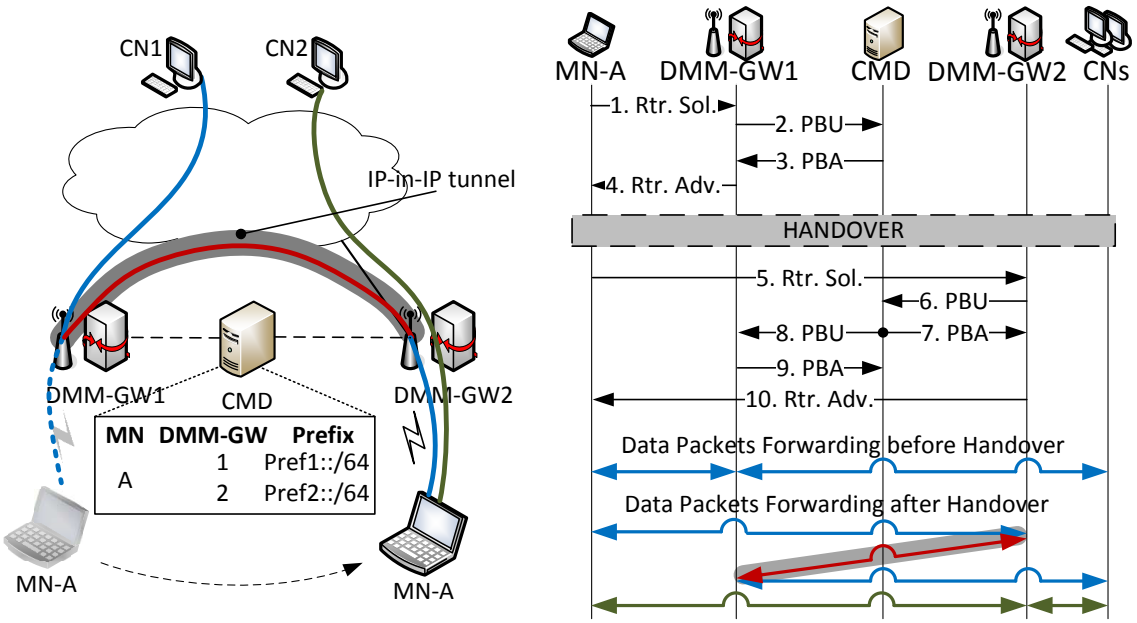


Figure 4.1: PMIPv6-based DMM solution.

the *anchoring* DMM-GW including the parameters from the new DMM-GW, and then receives the PBA from the old DMM-GW (9). By doing so, both the new and old DMM-GWs have the necessary information to set up a tunnel between them to recover the ongoing IP flows. The tunnel is used for those flows started before the MN handed over from the previous DMM-GW, whereas new communications are handled by the new DMM-GW as a plain router, i.e., without using any tunnels. This dynamic behaviour is achieved by the MN obtaining a new IPv6 prefix from each DMM-GW it connects to (10). Consequently, an MN configures several IPv6 addresses, one per each visited DMM-GW, and its flows might be anchored at different DMM-GWs. One of the main advantages of this approach is that new flows started when the node is attached to the new DMM-GW are not tunnelled, hence they do not suffer from any overhead or non optimal routing, improving the overall performance of the network.

The above solution extends PMIPv6 to provide a flatter mobility architecture, and therefore, it is based on the same principles. Its main advantage is that it is an evolved solution based on existing mechanisms, that could be easily deployed on currently rolled-out networks. However, operators are already moving towards *software networks*, which are more flexible and allow for faster and richer service deployments.

4.1.2 DMM design principles, SDN challenges and opportunities

This section highlights the main components of a DMM solution and the challenges introduced by shifting the architecture paradigm from pure IP to SDN. The SDN concept separates the control and the data forwarding planes. Such separation allows for quicker provisioning and configuration of network connections. This approach decouples the system making decisions about routing (i.e., control plane) from the underlying system that forwards traffic to the selected destination (i.e., data plane). In an SDN environment, the entity in charge of implementing the control logic for the network is called Network Controller (NC) and it is responsible of configuring the nodes in the network (data plane) via a common Application Programming Interface (API). A well-known SDN framework is the OpenFlow protocol and switch specification [ONF15a], which can be used by an external software application to program the data plane of network devices. SDN enables the partitioning of the control system into modular parts that can be dynamically composed according to the network needs. Nonetheless, the choice of component partitioning can have a profound influence on the types of services ultimately delivered to the end user [ONF16]. With this in mind, the following questions are formulated by departing from the DMM solution presented in the

previous section:

> What are the tasks to be accomplished by a generic DMM solution to efficiently provide MNs with mobility support?

> What are the challenges and opportunities in accomplishing these tasks following the SDN paradigm?

Generally speaking, each task can be seen as a stand-alone module that interacts with other modules in order to achieve MN's mobility. As a consequence, a generic DMM solution can be seen as a set of cooperating modules thus facilitating the evolution of such mobility solutions from IP architecture to SDN paradigm. These modules can be implemented and deployed in different ways. However, in order to provide a full-fledged mobility support to the MNs, they need to keep the same semantic interface towards the other modules. The following paragraphs describe the modules that a DMM solution should implement based on an analysis of the PMIPv6-based DMM solution.

Attachment detection

As shown in Figure 4.1, the whole PMIPv6-based DMM mobility procedure is triggered by a RS upon MN attachment (1) or handover (5). By generalising this concept, it can be argued that a DMM solution requires a specific module capable of detecting the MN's attachment or handover. Moreover, such a module should be provisioned with the following information to effectively trigger the mobility procedure: (*i*) the MN identity and, (*ii*) the DMM-GW the MN is attached to. Moving towards SDN, such information can be also retrieved by other means that do not strictly belong to the IP solution space. For example, in addition to *Layer-3* mechanisms, the attachment (or handover) can be also detected at *Layer-2* or via a *dedicated interface*. For instance, Logical Link Control (LLC) Subnetwork Access Protocol (SNAP) messages can be used for detecting new hosts in IEEE 802.1Q bridged networks [IEE03][23] while S1-MME dedicated interface can be used in 3GPP networks [3GP18h][24]. The possibility of employing different mechanisms, even simultaneously, however poses a significant challenge in the design of a SDN-based DMM solution: the MNs must be uniquely identified despite of the connectivity technology being used. In case of achieving such unique identification, a SDN-based DMM solution can potentially be extended across different technology domains thus providing an inter-technology mobility.

Binding

Upon successful attachment detection in PMIPv6, the DMM-GW notifies the CMD about the IPv6 prefix reserved for the MN through PBU messages (2,6). In practical terms, this operation can postulated as a stand-alone module providing a binding procedure that assigns one or more IPv6 prefixes to the MN. Moreover, the module keeps track of the assigned prefixes during the connection lifetime of the MNs. Therefore, two main tasks are accomplished by this module: (*i*) MN tracking and, (*ii*) prefixes selection. The tracking function keeps record of the whereabouts of the MN along with the IPv6 prefixes assigned by the prefix selection function. While in IP based solutions the tracking function usually stores only the visited DMM-GW, additional flexibility can be envisioned in SDN-based solution where supplementary localisation services, e.g. Global Positioning System (GPS), could be leveraged. Such information can be potentially taken into consideration next by the selection function which can select better anchor points for the MNs in combination with some network policies. From a deployment point of view,this function can be deployed in a *centralised* or *distributed* fashion. A *centralised* approach co-locates the selection and tracking functions, while a *distributed* approach separates them.

[23] IEEE. *Virtual Bridged Local Area Networks*. IEEE Standards for Local and metropolitan area networks 802.1Q. Institute of Electrical and Electronics Engineers (IEEE), May 2003.

[24] 3GPP. *S1 Application Protocol (S1AP)*. Technical Specification (TS) 36.413 v15.2.0. 3rd Generation Partnership Project (3GPP), June 2018.

Prefix advertisement

After the Binding function selects the prefix(es) to be assigned to the MN, the solution communicates them to the MN. For example, the PMIPv6-based DMM solution advertises such prefixes through RA messages (4,10) [Nar+07]. Alternatively, such communication may also occur via Dynamic Host Configuration Protocol (DHCP) Offer/DHCP Acknowledgement [Dro+03][25]. Upon RA reception, the MN configures its IP addresses starting from the assigned IP prefixes. It is worth highlighting that this chapter addresses network-based DMM solutions, therefore no direct interaction is envisioned between the mobility protocol and the MN, which in turn can also be unaware of any mobility support. As a consequence, a DMM network-based SDN solution can only leverage *Layer-3* mechanisms for advertising the prefix. This function can be deployed in a *centralised* or in a *distributed* fashion, based on the entity in charge of communicating the prefix to the MN. In case of a *centralised* function, a single module (e.g., the network controller) takes care of retrieving all the MN's assigned prefixes and to communicate them to the MN via a unique message. On the contrary, a different implementation may rely on standard mechanisms to provide the assigned prefixes to the MN, for example allowing each assigned router to advertise its own prefix to the MN in a *distributed* way.

Traffic steering

This module takes care of re-steering the MNs' traffic, thus effectively providing mobility. Upon an MN's attachment, the ongoing traffic must be re-routed to the new MN's location. This traffic path modification can be achieved in several ways, for instance, the PMIPv6-based DMM solution leverages on Layer-3 *tunnels*, i.e., IP-in-IP and/or the GTP [3GP18f][26]. An SDN-based solution can also envision the usage of other traffic steering techniques, such as *path reconfiguration*, or Virtual LAN (VLAN) [IEE03] /Multiprotocol Label Switching (MPLS) [RVC01][27] *tagging*. This module may implement the whole function taking care of physically implementing it, or, it may interact with an external module or entity that is in charge of the configuration, e.g., a Path Computation Element (PCE) or a dedicated module on the NC. In the latter case, the traffic steering module communicates the paths that must be configured to the external module (i.e., a path from the MN to the DMM-GWs and vice-versa). Subsequently, the external module will configure the underlying network accordingly. It is worth highlighting that the SDN architecture defined in [ONF16] enables SDN modules, like the ones described above, to interact with the NC and request the configuration of some network paths. The NC has hence the responsibility to enforce the request in the underlying network, which may involve a combination of traffic steering techniques depending on the network deployment. For example, MN traffic steering can be a combination of Layer-3 tunnels and VLAN tagging across different network segments. Despite of the added complexity in the SDN approach, a careful combination of steering techniques can lead to a better usage of the resources due to a increased path optimality that can be potentially achieved in the network.

From the perspective of an SDN architecture, the modules described above can be implemented as applications running on top of one or multiple SDN controllers. The choice of using one or multiple SDN controller depends on the design adopted for the solution. So far, we have only described the functionality required for a correct operation of a DMM solution. Nevertheless, the modules must cooperate and interact with each other. In order to do so, the specific control and data planes used for the mobility solution[28] need to be defined.

[25] R. Droms et al. *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. Request for Comments (RFC) 3315. Internet Engineering Task Force (IETF), July 2003.

[26] 3GPP. *General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)*. Technical Specification (TS) 29.281 v15.3.0. 3rd Generation Partnership Project (3GPP), June 2018.

[27] E. Rosen, A. Viswanathan and R. Callon. *Multiprotocol Label Switching Architecture*. Request for Comments (RFC) 3031. Internet Engineering Task Force (IETF), January 2001.

[28] Note that these control and data planes refer only to mobility specific functionality and are different from the traditional control and data split considered in SDN approaches.

Mobility control plane

It is the control plane adopted by the mobility solution and it can be *dedicated* or *compliant*. In case of a *dedicated* control plane, the mobility solution employs a different control plane for the signalling with respect to the one used by the SDN controller (e.g., based on PMIPv6 or GTP-C [3GP18c][29]). On the contrary, in case of a *compliant* control plane, the mobility solution uses the same southbound signalling used by the SDN controller (i.e., OpenFlow). A mobility solution may use a *mixed* control plane. For example, the attachment detection may employ the OpenFlow PacketIn event, while the communication between different modules leverages PBU/PBA messages.

Mobility data plane

It can be *dedicated* or *shared*. A *dedicated* data plane implies that the packets exchanged between the MN and the DMM-GW (i.e., Address Resolution Protocol (ARP), DHCP or IPv6 Neighbour Discovery) may follow a different path with regard to the packets belonging to the MN's data plane. In case of *shared* data plane, the two data planes are not separated.

4.2 An SDN-based DMM solution

This section describes in details the proposed SDN-based DMM solution, which will be later evaluated and compared to the PMIPv6-based one introduced before. As previously commented, designing an SDN-based DMM solution requires additional efforts with respect to a classic IP mobility solution where it is safe to assume that every node in the network speaks IP. However, this cannot be assumed in an SDN environment as introduced in the previous section. If we consider the more generic SDN concept – that is the decoupling of control and data planes – several boxes are controlled remotely by a NC in order to accomplish a complex task in the network. In such scenario, the NC controls and instructs the data plane nodes via a *southbound interface* which defines the instruction set understandable by both the NC and the data plane nodes. Although there are multiple possibilities for the choice of Southbound interface, we have decided to use the OpenFlow protocol, which enables the NC to write forwarding rules directly on the nodes. Given the nature of this paper, which evaluates the proposed DMM solutions from analytic and experimental viewpoints, our SDN-based mobility focuses on standard OpenFlow v1.5 capabilities, leaving potential extensions and non-standard Southbound interfaces out of the scope of this work. Undeniably, various Southbound-API may accomplish the same task in different ways but there are cases where one task cannot be accomplished by a specific Southbound-API. For example, let's analyse the classical IP mobility mechanisms and OpenFlow. Classical IP mobility solutions (i.e., [Gun+08], [Per02][30], [PJA11]) exploit IP tunnels for re-routing the traffic. Unfortunately, even the latest OpenFlow specification [ONF15a] does not include any instruction for managing IP tunnels[31]. Hence, in OpenFlow networks, mobility can be only supported by changing the forwarding rules in the data plane. According to the *Traffic steering* module described in Chapter 4.1.2, and to OpenFlow specifications, traffic steering can be only done via path reconfiguration or VLAN in an OpenFlow network.

In the proposed solution, the modules introduced during the discussion on the DMM design principles can be mapped to an SDN paradigm as different applications running on top of one or more NCs. Therefore, the DMM-GWs are designed as plain forwarding nodes (switches), bearing no mobility functionality, thence delegating the whole intelligence to the NC. The applications adopt an event-driven communication paradigm in order to be as modular and reactive as possible, being this aspect fundamental in mobility solutions. Indeed, an event-driven communication can

[29] 3GPP. *Evolved General Packet Radio Service (GPRS) Tunnelling Protocol for Control plane (GTPv2-C)*. Technical Specification (TS) 29.274 v15.4.0. 3rd Generation Partnership Project (3GPP), June 2018.

[30] C. Perkins. *IP Mobility Support for IPv4*. Request for Comments (RFC) 3344. Internet Engineering Task Force (IETF), August 2002.

[31] Note there are some vendor specific extensions able to control some kinds of tunnelling, e.g., GRE tunnels.

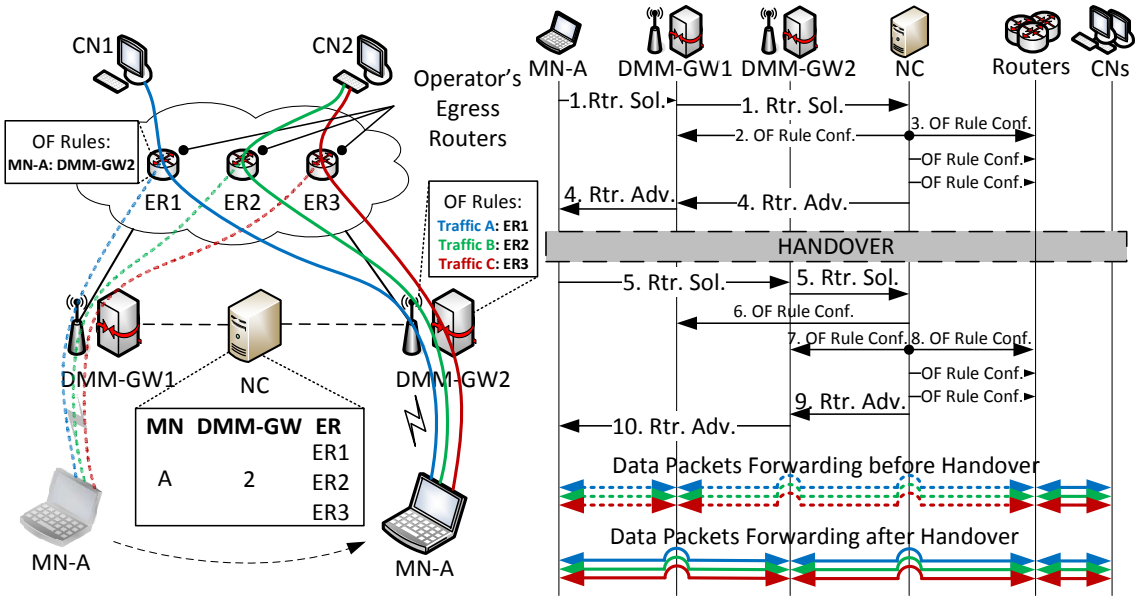


Figure 4.2: SDN-based DMM solution.

Table 4.1: SDN-based DMM solution modules details.

	Deployment	Data required	Event required	Event provided
(1)	Layer 3	Router Solicitation	OF PacketIn	MN attachment
(2)	Centralised	MN IDs	MN attachment	Binding update
(3)	Centralised	Binding cache	Binding update	-
(4)	VLAN	Binding cache	Binding update	-

(1) Attachment; (2) Binding; (3) Prefix Advertisement; (4) Traffic Steering;

be used to fire triggers upon certain events, like the mobility support being activated by an MN handover.

As shown in Figure 4.2, upon the attachment of an Mobile Node (MN) to an access point, the DMM-GW informs the Network Controller (NC), which assigns a network prefix (or a set of prefixes, in case differentiated treatment is required for fine-grained services) to the MN. The network prefix(es) is guaranteed to be unique by using a binding cache where the controller stores information about the MNs active in the network (and the prefixes they use). The detection of the attachment and the network prefix assignment in the proposed solution, is based on IPv6 Neighbour Discovery as in the PMIPv6-based solution: the MN sends a Router Solicitation (RS) when attaches to the network (1,5), that serves as trigger, and the NC generates a Router Advertisement (RA) to communicate the network prefix(es) (4,10). These prefixes are anchored at a pool of k Egress Router (ER). After the selection and assignment of IP prefixes (and associated ERs), the NC configures the OpenFlow rules in the MN's target DMM-GW (2,7) and in the ERs assigned to it (3,8). In case of handover, the NC also deletes the OpenFlow rules previously installed on the old DMM-GW (6). Packet forwarding within the network is based on VLANs, which are statically pre-configured. Such VLAN paths connect Egress Routers with DMM-GWs. Note that these VLAN paths could also be dynamically configured by the NC using OpenFlow, but this procedure works in a different time scale with respect to mobility management and is ruled out of the scope of this work. As a matter of fact, packet re-routing could be also based on path reconfiguration. However, re-configuring the whole path leads to a higher and non-deterministic signalling load due to the variable length of the paths in the network. Mobility support is achieved by installing OpenFlow rules at the Egress Routers and DMM-GWs, so packets destined or originated from an MN are tagged with the correct VLAN (see Figure 4.2). Upon the attachment of an MN to the network, the NC configures Egress Routers to tag MN's packets with the VLAN connecting the

Table 4.2: Distributed Mobility Management notation.

SYMBOL	DESCRIPTION	SYMBOL	DESCRIPTION
MN	Mobile Node	$G^A \subseteq G$	Set of active DMM-GWs at each handover
DMM-GW	DMM Gateway	$g_i^A \in G^A$	Element of set G^A
ER	Egress Router	$c\left(g_i^A\right)$	Signalling cost associated to node g_i^A
CMD	Control Mobility Database	C_h	Total handover signalling cost
NC	Network Controller	E	Set of Egress Routers
RS	An IPv6 Router Solicitation message	$e_m \in E$	Element of set E
RA	An IPv6 Router Advertisement message	S_X	Entity X's forwarding table size (#rules)
RTT	Round Trip Time	U	Set of MNs connected to the domain
G	Set of DMM-GWs	$U^{g_i \subseteq U}$	Set of MNs associated to DMM-GW g_i
$g_j \in G$	Element of set G	$U^{e_m} \subseteq U$	Set of MNs associated to ER e_m

Egress Router with the DMM-GW the MN is attached to. In case of handover, the NC simply needs to rewrite this rule at the Egress Router and the DMM-GW, selecting the correct VLAN that connects the new DMM-GW and the ERs assigned to the MN.

Summarising, the proposed solution envisions a shared data plane and a compliant control plane, exploiting OpenFlow as signalling protocol. In addition, Table 4.1 reports the deployment, the data and events required by each module to properly work and interact. For example, the attachment detection occurs at Layer-3 by intercepting the Router Solicitation message and sending it to the NC as the payload of an OpenFlow PacketIn message. Such module broadcasts an *MN attachment* event to the other modules which contains the MN's ID and point of attachment. This event is received by the centrally-deployed Binding module which selects the MN's network prefixes and update accordingly the binding cache. Once the binding cache is correctly updated, the module broadcasts a *Binding up* message which includes the MN's prefixes and associated DMM-GWs. Such event is exploited by (*i*) the centralised prefix advertisement module to send a RA to the MN, and by (*ii*) the traffic steering module to update the VLANs tag in the DMM-GWs and ERs.

4.3 Analytic evaluation

In this section, the PMIPv6 and SDN-based solutions are analysed considering the signalling overhead and the overall handover latency. We further analyse the scalability of the proposed solutions in terms of size of the forwarding tables which has an impact on the state that network nodes need to keep to properly operate. Table 4.2 summarises the notation used throughout this section.

4.3.1 Signalling cost

The signalling cost is formulated as the overhead in bytes associated to the control messages transmitted when an MN performs a handover. The purpose of this study is to provide a statistical estimation of the signalling rate in B/s, based on mobility and traffic models available in the scientific literature, such as the analytical framework proposed in [GBO14], which properly captures the peculiarities of DMM protocols. The formal definition of the cost model is expressed in the following.

Proposition 4.1: Let G be the set of DMM-GWs deployed in a given area, and $g_i \in G$ its elements.

When a mobile node hands off, only few DMM-GWs are involved in the signalling process, therefore:

Proposition 4.2: Let $G^A = \{g_1^A, \ldots, g_N^A\}$ be the set of *active DMM-GWs* participating in a handover control sequence.

Clearly, $G^A \subseteq G$, and $N = \left|G^A\right|$. The set G^A changes at each handover, and its elements g_i^A are reverse-ordered from the latest to the first active DMM-GW visited by the MN. Thence, a_1

Table 4.3: DMM signalling messages cost.

PACKET	BYTES	DESCRIPTION
π_{PBA}	128	PBA with mandatory options only
$\pi^{\text{option}}_{anchor}$	56	Previous DMM-GW mobility option
$\pi^{\text{option}}_{serving}$	24	Current DMM-GW mobility option
σ_{RS}	178	Router Solicitation sent to the NC
σ_{RA}	218	Router Advertisement sent by the NC
σ_{write}	264	OpenFlow message for writing a rule
σ_{delete}	232	OpenFlow message for deleting a rule

is the hand-off target DMM-GW and a_2 is the source DMM-GW. It should be noted that, in the PMIPv6-based solution, the size N varies at each handover, whereas for the SDN solution we have $N = 2$ for each handover. We now characterise the handover cost in terms of signalling load.

Proposition 4.3: Let $c\left(g_i^A\right) : G^A \mapsto \mathbb{N}$ be the cost in bytes of each information exchange for any given $g_i^A \in G^A$ that involves g_i^A, including the IPv6 and transport-layer headers, but excluding the data link and MAC layer headers.

Therefore, the handover cost is modelled as follows:

Definition 4.1: Handover cost.

$$C_h = \sum_{g_i^A \in G^A} c\left(g_i^A\right), \tag{4.1}$$

whereas the function $c\left(g_i^A\right)$ is solution-dependent.

The characterisation of $c\left(g_i^A\right)$ is addressed in the following paragraphs.

PMIPv6-based

To properly formalise $c\left(g_i^A\right)$, some handover operations are detailed. The target DMM transmits a PBU message with the PMIPv6 mandatory options only (denoted as π_{PBU}) to the CMD to notify the MN's new attachment. The CMD replies with a PBA including the mandatory options (π_{PBA}) plus an instance of the *anchor option* ($\pi^{\text{option}}_{anchor}$) for every old DMM-GW that is still anchoring active IP flows. Similarly, the source DMM-GW, and all the other DMM-GWs that are still anchoring IP flows, receive from the CMD a PBU message with the mandatory options plus an instance of the *serving option* ($\pi^{\text{option}}_{serving}$), indicating the new serving DMM-GW. These DMM-GWs then reply to the CMD with a PBA containing the same options to conclude the operation.

Proposition 4.4: Let the signalling cost for the PMIPv6-based DMM solution be:

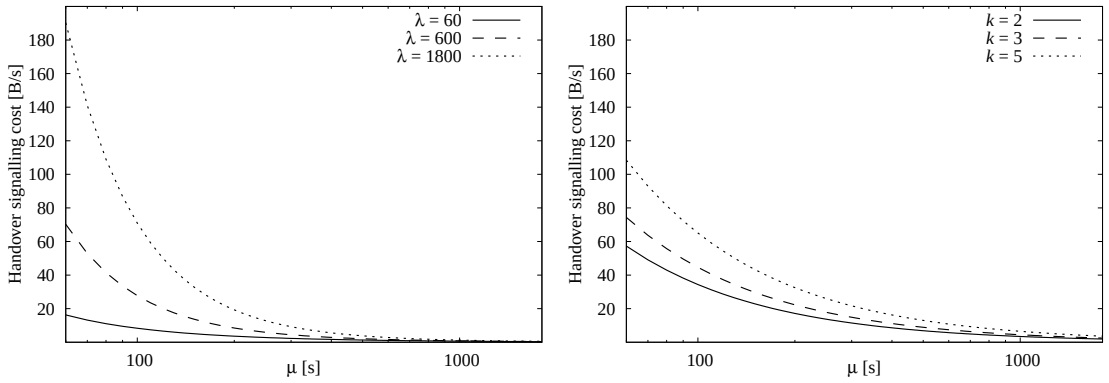
$$c\left(g_i^A\right) = \begin{cases} \pi_{PBU} + \pi_{PBA} + (N-1)\pi^{\text{option}}_{anchor} & \text{if } i = 1 \\ \pi_{PBU} + \pi^{\text{option}}_{serving} + \pi_{PBA} + \pi^{\text{option}}_{serving} & \text{if } i \geq 2 \end{cases} \tag{4.2}$$

Therefore, the Equation 4.1 turns into Equation 4.3 for the PMIPv6-based case:

Definition 4.2: Handover cost of the PMIPv6-based DMM.

$$C_h^{\text{PMIPv6-based}} = N\left(\pi_{PBU} + \pi_{PBA}\right) + (N-1)\left(\pi^{\text{option}}_{anchor} + 2\pi^{\text{option}}_{serving}\right) \tag{4.3}$$

In conclusion, this solution's cost depends linearly on the number $N = \left|G^A\right|$ of active DMM-GWs. The value of N depends on both the MN mobility (i.e., handover frequency) and traffic patterns. It is intuitive that the more often the MN changes attachment point, the larger is the number of active DMM-GWs. However, a DMM-GW is eventually de-activated when there are no more MN's IP flows traversing it. So, the longer the IP flows started by the MN are, the longer is the DMM-GW's activity interval. Knowing the statistical distribution of the handover rate and



(a) PMIPv6-based solution (b) SDN-based solution

Figure 4.3: Handover signalling cost for the two DMM solutions.

how long an IP flow is maintained by the DMM-GW anchoring that flow permits to compute the statistical distribution of the number of active DMM-GWs at any time [GBO14], and thus the size of the set G^A. In this chapter we simplify the problem assuming that an MN spends an exponential time with mean value μ attached to a DMM-GW before handing over to a different one. Besides, we assume that after a handover, an old DMM-GW remains active for an exponential interval of mean λ. Using the results reported in [GBO14], we obtain $\overline{N} = \mathrm{E}[N]$, as:

Definition 4.3: Average number of active DMM-GWs.

$$\overline{N} = 2 + \frac{\lambda}{\mu} \tag{4.4}$$

SDN-based

In the SDN-based solution, the only DMM-GWs involved during a handover are the source and target DMM-GWs, thus $G^A = \{G_1^A, G_2^A\}$ for every handover. From the protocol description in Chapter 4.2, upon a handover, the NC writes on the target DMM-GW two downlink rules and as many uplink rules as the number of active ERs for the mobile node. Moreover, the NC writes two downlink rules on each ER, and removes the uplink and downlink rules on the source DMM-GW. The reason of having two downlink rules is given by the need of properly identifying the mobile node's IPv6 local and global addresses[32]. Thus, for the general case of having k ERs, and using the message notation shown in Table 4.3, the cost function is defined as follows:

Proposition 4.5: Let the signalling cost for the SDN-based DMM solution be:

$$c\left(g_i^A\right) = \begin{cases} \sigma_{RS} + \sigma_{RA} + (3k+2)\,\sigma_{write} & \text{if } i = 1 \\ (k+2)\,\sigma_{delete} & \text{if } i = 2 \end{cases} \tag{4.5}$$

Therefore, Equation 4.1 turns into Equation 4.6 for the SDN-based case:

Definition 4.4: Handover cost of the SDN-based DMM.

$$C_h^{\text{SDN-based}} = \sigma_{RS} + \sigma_{RA} + (3k+2)\,\sigma_{write} + (k+2)\,\sigma_{delete} \tag{4.6}$$

As it can be observed in Equation 4.6, the SDN solution's cost depends linearly on k.

Signalling cost considerations

The average signalling cost for a single MN is analyzed in the following for the two solutions. The average residence time μ is the one defined previously for the PMIPv6-based solution, and C_h/μ

[32] IPv6 uses unicast and multicast addresses to reach the mobile node. Even if the identification would be based on MAC addresses, it would still require two rules.

is the corresponding solution's cost in bytes per second. The size of each message involved in Equations 4.2,4.5 has been measured experimentally (and its value is reported in Table 4.3). The description of the experiments is reported later in Chapter 5.

The signalling cost is reported, for different values of λ and k, in Figure 4.3(a) for the PMIPv6-based solution and in Figure 4.3(b) for the SDN-based. A performance degradation is observed on the PMIPv6-based solution for large values of the ratio λ/μ values, and hence large $\overline{N}$. This is a scenario with high mobility and long lived IP flows. In order to cope with this limitation, the deployment of such solution should jointly consider the coverage area and the level of mobility of MNs. That is, the DMM-GW's coverage area should not be too small in order to reduce the number of active DMM-GWs. Such information is usually available to operators. Nevertheless, this solution is more suitable when handling scenarios with low mobility, i.e., for high values of μ, or short lived flows, i.e., for low λ. The SDN-based solution behaves in a more predictable way, as it only depends on the value k and it is independent of the traffic pattern of MNs. Operators have the profiles of each MN, therefore the value k can be also adapted on an MN basis. As a result, the network can be managed in a smarter way and a higher network's efficiency can be achieved by spreading the MNs on multiple ERs.

4.3.2 Forwarding table size

The following characterises the parameter S as the size, i.e., the number of rules, of the forwarding table of the involved network nodes. The number of forwarding entries represents the state that each node needs to keep in the network to properly forward the traffic to the MNs. To that end:

Proposition 4.6: Let U be the set of MNs in the domain and $U^{g_i} \subseteq U$ be the set of MNs connected to a generic DMM-GW g_i.

Proposition 4.7: Let the MNs be uniformly distributed among the DMM-GWs.

Therefore, the number of MNs connected to a given DMM-GW is given by:

Definition 4.5: Average number of MNs connected to a DMM-GW.

$$|U^{g_i}| = |U| / |G| \tag{4.7}$$

where G is the set of DMM-GWs in the domain.

PMIPv6-based

As described in 4.3.1 and according with the statistical framework defined in [GBO14] an MN has on average $\overline{N}-1$ act an MN has $\overline{N}-1$ active prefixes advertised by old DMM-GWs, for which three routing rules are necessary: one at the anchor DMM-GW for downlink forwarding, and two at the current DMM-GW, respectively for uplink and downlink. In addition, the MN configures an IPv6 prefix from the current DMM-GW's pool, implying one downlink routing rule at the current DMM-GW. By dividing the total number of routing rules for the number of DMM-GWs, we obtain that, on average, the number of routing entries in a DMM-GW is given by:

Definition 4.6: DMM-GW forwarding table size for PMIPv6-based DMM.

$$S_{g_i}^{\text{PMIPv6-based}} = \left(3\overline{N}-2\right)|U^{g_i}| \tag{4.8}$$

SDN-based

For the SDN-based solution, the study applies to the DMM-GWs and ERs, as they are the only involved nodes.

Proposition 4.8: Let E be the set of egress routers deployed in a domain and $e_m \in E$ its elements.

While a DMM-GW manages no more than the U^{g_i} MNs directly attached to it, an ER manages the traffic of MNs that might be connected to multiple DMM-GWs. As described in Chapter 4.3.1, on each ER the NC writes two OpenFlow rules for each MN. Therefore, the forwarding table's size

on the m-th ER, e_m, is independent of k and depends only on the number of MNs managed by that ER. We denoted this set as $U^{e_m} \subseteq U$. As a result, the size of the forwarding table turns into:

Definition 4.7: ER forwarding table size for SDN-based DMM.

$$S_{e_m}^{\text{SDN-based}} = 2\,|U^{e_m}| \tag{4.9}$$

It is worth highlighting that this is the best achievable result. In fact, the smallest number of rules necessary to properly identify a single MN is two rules as explained in the previous section. On the contrary, the forwarding table on a DMM-GW depends on k and on U^{g_i}. Indeed, the NC writes $k+2$ rules on the target DMM-GW for each associated MN, leading to:

Definition 4.8: DMM-GW forwarding table size for SDN-based DMM.

$$S_{g_i}^{\text{SDN-based}} = (k+2)\,|U_{g_i}| \tag{4.10}$$

Proposition 4.9: Regarding U^{e_m} and U^{g_i}, we can safely assume:

$$|U| \geq |U^{e_m}| \gg |U^{g_i}| \qquad \forall i,m \tag{4.11}$$

That is the number of MNs managed by an ER is much larger than the number of MNs managed by a single DMM-GW (up to all the MNs in the domain). Therefore, k does not present a major scalability problem for the DMM-GW's forwarding table size.

4.3.3 Handover latency

This section analyses how the protocol operations affect the handover latency for each of the two studied DMM solutions. The handover delay analysis can be split into three sub-problems:(*i*) the Layer-2 handover, including the time elapsed since the old radio link is torn down until the new one is established, (*ii*) the Layer-3 configuration, considering the time required by the MN to obtain network layer connectivity (including the Layer-2 handover), and (*iii*) the IP flow recovery, i.e., the interval during which an IP flow is interrupted due to the handover (including both the Layer-2, the Layer-3 configuration, plus then the remaining actions performed within the network to ensure IP session continuity). In the protocols under consideration, the Layer-2 handover does not depend on the specific solution and it is the same for all of them, thus we omit it in the equations. Nevertheless, in Chapter 5 Layer-2 results obtained in our experiments are presented.

PMIPv6-based

The MN establishes the Layer-3 connectivity by requesting an IPv6 prefix with a RS message. The DMM-GW, before sending to the mobile node the IPv6 prefix information in a RA, performs a two-way message exchange with the CMD to register the MN presence and the assigned prefix. The message from the CMD contains the necessary parameters to set up the tunnels and the routing towards the old DMM-GWs that are anchoring the MN's active prefixes. As a result, the time required by the MN for the Layer-3 configuration is due to the Round Trip Time (RTT) between the MN and the DMM-GW for the RS/RA exchange, plus the RTT between the DMM-GW and the CMD for PBU/PBA signalling and, finally, a processing time $T_P^{\text{PMIPv6-based}}$ for each of the N active DMM-GWs. This Layer-3 latency can then be expressed as:

Definition 4.9: Handover latency of the PMIPv6-based DMM.

$$T_{L3}^{\text{PMIPv6-based}} = RTT_{MN\text{-}DMMGW} + RTT_{DMMGW\text{-}CMD} + NT_P^{\text{PMIPv6-based}} \tag{4.12}$$

In order to recover the IP flows started with the IPv6 prefixes assigned by previous DMM-GWs, the CMD instructs all the previous active DMM-GWs with parallel PBU/PBA signalling after the attachment notification from the new DMM-GW is received. For the model, it can be assumed that $RTT_{DMMGW\text{-}CMD}$ is constant for all the DMM-GWs, so that the new DMM-GW and all the old ones receive the update message from the CMD simultaneously. Next, an old DMM-GW re-builds the

data path with a tunnel to the current MN's DMM-GW in a time $T_P^{\text{PMIPv6-based}}$, and then packets flow to the serving DMM-GW in a time $(1/2)\,RTT_{DMMGW\text{-}DMMGW}$. Therefore, the flow recovery time is:

Definition 4.10: Flow recovery time of the PMIPv6-based DMM.

$$\begin{aligned} T_{flow\text{-}recovery}^{\text{PMIPv6-based}} &= T_{L2\text{-}ho} + RTT_{MN\text{-}DMMGW} + RTT_{DMMGW\text{-}CMD} + \\ &+ T_P^{\text{PMIPv6-based}} + \tfrac{1}{2}\,RTT_{DMMGW\text{-}DMMGW} \end{aligned} \tag{4.13}$$

SDN-based

In this case, RS sent by the mobile node upon attachment is intercepted by the target DMM-GW and forwarded to the NC. At this point, the NC configures the forwarding path in the network by: (*i*) writing the rules on the new DMM-GW and on the ERs, and, (*ii*) deleting the rules on the old DMM-GW. In this solution, the order plays an important role, indeed, after the configuration of the target DMM-GW and of the ERs, the path in the network is updated and the traffic is finally able to reach the MN. Consequently, to compute the IP flow-recovery time, we can get rid of the time necessary to delete the rules on the last visited DMM-GW. After the configuration phase, the NC generates an RA which is sent back to the DMM-GW and finally forwarded by the latter to the MN. This RS/RA exchange lasts an RTT between the MN and the DMM-GW plus another RTT between the DMM-GW and the NC. In the new DMM-GW, the NC writes $k+2$ rules through $k+2$ parallel messages, where k is the number of ERs. As a result, the NC takes $(1/2)\,RTT_{DMMGW\text{-}NC}$ to configure the new DMM-GW. For the ERs, the NC writes two rules on each ER in parallel, taking $(1/2)\,RTT_{ER\text{-}NC}$ to update the rules, where $RTT_{ER\text{-}NC}$ is the distance between the ER and the NC. For simplicity, we consider $RTT_{ER\text{-}NC}$ as constant for each ER. Albeit the NC configures the rules in parallel, the generation of those messages is performed sequentially. $T_P^{\text{SDN-based}}$ denotes the processing time required by the NC to forge the OpenFlow messages for each of the k ERs. Therefore, the Layer-3 configuration latency is:

Definition 4.11: Handover latency of the SDN-based DMM.

$$\begin{aligned} T_{L3}^{\text{SDN-based}} &= RTT_{MN\text{-}DMMGW} + \tfrac{3}{2} RTT_{DMMGW\text{-}NC} + \\ &+ \tfrac{1}{2} RTT_{ER\text{-}NC} + k\,T_P^{\text{SDN-based}} \end{aligned} \tag{4.14}$$

After the configuration phase and the reception of the RA message by the MN, the packets can finally reach the MN, taking a time $T_{transport}$. Hence, the flow-recovery time is:

Definition 4.12: Flow recovery time of the SDN-based DMM.

$$T_{flow\text{-}recovery}^{\text{SDN-based}} = T_{L2\text{-}ho} + T_{L3}^{\text{SDN-based}} + T_{transport} \tag{4.15}$$

In addition to the analytic evaluation of the handover latency presented in this chapter, an experimental evaluation of both DMM solutions (i.e., PMIPv6 and SDN) is presented in Chapter 5.

5. Experimental assessment of the SDN framework

The main goal of this chapter is to provide evidence in a real-life environment of the benefits of applying Software Defined Networking (SDN) concepts to mobile networks in terms of easiness, flexibility and agility when deploying new services. To do so, an SDN test-bed has been developed based on off-the-shelf hardware running GNU/Linux.

5.1 Evaluation of service creation effort

The following analyses the implementation of the proposed SDN framework (see Chapter 3) to quantify the actual effort required for creating SDN controller modules and applications. Even though it is generally claimed that reducing service creation time is one of the key advantages of SDN, to the best of the author's knowledge, this is the first attempt to quantify such benefit. The effort required to implement a new service depends on the complexity of the task that the service aims to accomplish, and the ease with which the platform that implements it can be used. While the complexity of the task is determined by the use case, the implementation effort is highly influenced by the tools offered to the developers. The goal of this section is therefore to evaluate the developer-friendliness of the proposed framework. In particular, this section quantifies the implementation effort associated to the exemplary use cases reported in Chapter 2. The network controller runs Ryu[1] as component-based SDN framework. Ryu APIs natively support the event-driven communication paradigm allowing a simpler prototyping of the proposed architecture. Moreover, the Topology Discovery module is already provided by Ryu. All the other modules of the architecture have been implemented using Python. The next paragraphs evaluate the Controller and Application Plane implementations efforts.

Table 5.1 reports the *Implementation effort*, both in terms of lines of code and the development time spent. Those figures are of course specific to the implementation choices that have been made (e.g., the use of the Ryu controller and the Python language) and to the developer, whose skills are reported in Table 5.2 in the form of a programmer competency matrix[2] which is commonly used for assessing a specialist's competences [Lyt+16][3]. These provide a valuable insight on the complexity of the proposed architecture and an good estimation of the required effort, although there might be differences across developers. The architecture components are divided in two groups: the core *Controller* modules, which provide the basic functionality of the architecture, and the *New services*

[1] Ryu: `http://osrg.github.io/ryu/`

[2] Competency matrix: `http://sijinjoseph.com/programmer-competency-matrix/`

[3] V. Lytvyn et al. 'A method for constructing recruitment rules based on the analysis of a specialist's competences'. In: *Eastern-European Journal of Enterprise Technologies* 6.2 (December 2016), pages 4–14. DOI: 10.15587/1729-4061.2016.85454.

Table 5.1: Evaluation of implementation effort.

MODULE	LINES OF CODE	TIME SPENT
Controller Modules	3218	95 hours
New Services	1589	20 hours 40 minutes
Mobility Management	356	4 hours
Privacy	153	40 minutes
Traffic Engineering	192	1 hour
Service Function Chaining	216	1 hour
Multi Tenancy	672	14 hours

Table 5.2: Programmer competency matrix.

FIELD	SKILL	LEVEL 0	1	2	3
Computer Science	Data structures			X	
	Algorithms			X	
	System programming				X
Software Engineering	Build automation			X	
	Automated testing			X	
Programming	Problem decomposition				X
	System decomposition				X
	Code readability			X	
	Defensive code			X	
	Error handling			X	
	API design				X
	Framework design			X	
	Requirements definition				X
Experience	Languages experience			X	
	Platforms experience			X	
	Domain experience			X	
Knowledge	Tools knowledge			X	
	Upcoming technologies				X

modules, which build on the former to implement the advanced functionality of the network. As it can be seen in the table, the core modules require a relatively large implementation cost (almost 100 hours); however, this cost is incurred only once. In contrast, the application involve a much lower cost (approx. one fifth of the core modules for all the considered functionality). In general terms, this shows that the proposed Open Networking Foundation (ONF)-based architecture enables network operators to provision new services with a very low service creation time. The following summarises the implementation challenges of these modules.

The implementation of the *New Services* modules required relatively little time. The Multi Tenancy module was the most challenging application to implement (more than two thirds of the overall effort): while the implementation of its interface was straightforward and its engine to ensure consistency is relatively simple, significant time was devoted to its testing and validation to proper handling concurrent requests. The implementation of the other services required much less time: Mobility Management[4] and Privacy modules required less than 5 hours and basically consist of a smart algorithm for gateway selection, while Traffic Engineering module, which is based on a solver for the linear programming formulation,[5] required one hour (i.e., one twentieth of the total effort implementing new services). The implementation of the Service Function Chaining is another key feature of the implementation, as it provides the network operator with an interface for defining and modifying paths between network elements, requiring a similar implementation

[4] This module has been published as open-source and it is available for download at: `http://odmm.net/openflow/`

[5] SciPy.org, Linear Programming Solver: `https://docs.scipy.org/`

effort.

To support the development of the above modules, continuous integration (CI) tools were used for streamlining the code testing and debugging. For automatic testing a local installation based on Docker[6] of Travis CI was used,[7] providing a customizable service for building and testing software projects. A series of unit tests are hence executed to (*i*) verify the correctness of the modules implementation against the expected behaviour of the events and API, as defined in Table 3.1 and 3.2, and to (*ii*) attest the robustness of the implementation against unexpected inputs or events. Moreover, Codecov[8] was used for assessing the coverage of the unit tests on the implemented code as to ensure that any line being developed is properly tested. While those tools provided the necessary testing functionalities in the context of this evaluation, the use of a more complete suite, like the one proposed in [Gut+16][9], is advisable to better automatise the development process and thoroughly detect undesirable bugs, as highlighted in [JM16][10]. For the sake of clarity, all the controller modules evaluated in this article have been implemented on a single controller. However, regardless the implementation of each controller module, the interface toward the Application plane remains the same and does not require any change on the applications thanks to the adoption of a service-oriented architecture. Indeed, one of the key benefits of such architecture is that interactions occur between loosely coupled software components that operate independently. Moreover, this architecture allows for service reuse, making it unnecessary to rewrite all the components when upgrades/modifications are needed only affect a subset of the modules. As a result, the evaluation of the effort required to implement the applications is still accurate although being developed on a proof-of-concept.

Notwithstanding, it is worth noticing that the implemented prototype faces the same challenges of centralised systems, in terms of reliability, resiliency and scalability. Because of these issues, a deployment of the controller in a realistic scenario might be distributed across a number of separate servers. Therefore, each controller module could be implemented in a distributed fashion relying on well-known high performance distributed computing (HPDC) techniques. For example, the Topology Discovery and Topology View modules can be implemented using a divide and conquer approach where the whole network domain is split in sub-domains and the global network view is created by combining all the partial views. Path Computation might follow the same approach whereas the computed paths are exposed via a distributed hash table (DHT). In this way, the module can scale to large number of paths and to handle continuous updates and requests. Furthermore, additional applications can be implemented in order to support and inter-operate with legacy modules (e.g., ANDSF, HSS, AAA, IMS) following the same approach of this implementation.

5.2 Evaluation of DMM solutions

Complementing the analytical evaluation conducted in Chapter 4, this section describes the experimental evaluation of the PMIPv6-based and the SDN-based Distributed Mobility Management (DMM) solution, aiming at providing a Proof of Concept (PoC) to assess the solutions' feasibility and performance. In order to evaluate the two solutions, two separate test-beds are developed, one for each solution. Each test-bed is based on GNU/Linux machines connected through an Ethernet network and comprises a set DMM Gateway (DMM-GW)s providing Institute of Electrical and Electronics Engineers (IEEE) 802.11b/g wireless access to the Mobile Node (MN)s. The systems are tested for three different configurations, employing 2, 3 or 5 DMM-GWs. As none of the

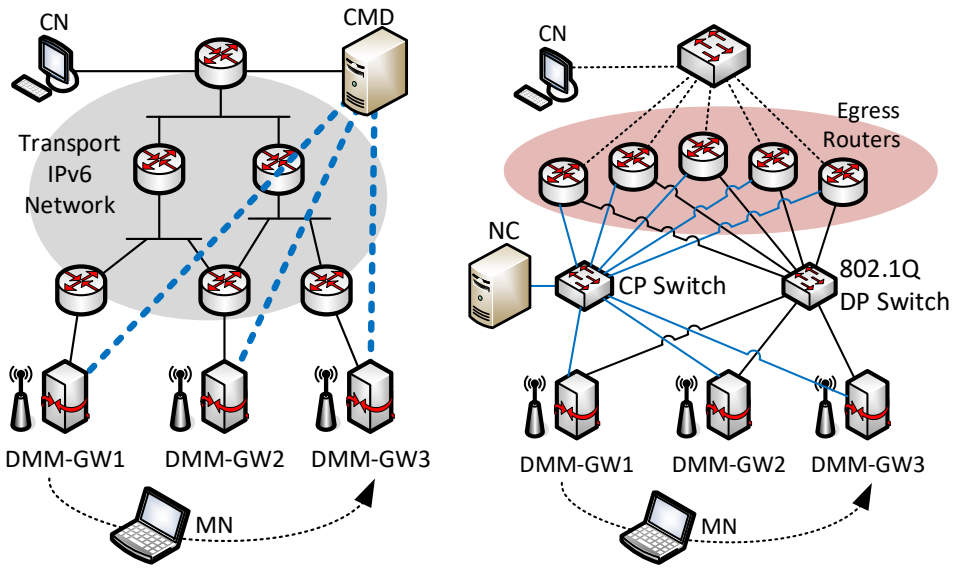
[6] Docker: `https://www.docker.com/`

[7] Tracis CI: `https://travis-ci.org/`

[8] Codecov: `https://codecov.io/`

[9] P. A. A. Gutiérrez et al. 'NetIDE: All-in-one framework for next generation, composed SDN applications'. In: *2016 IEEE NetSoft Conference and Workshops (NetSoft)*. June 2016, pages 355–356. DOI: `10.1109/NETSOFT.2016.7502408`.

[10] L. J. Jagadeesan and V. Mendiratta. 'Programming the Network: Application Software Faults in Software-Defined Networks'. In: *2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. October 2016, pages 125–131. DOI: `10.1109/ISSREW.2016.23`.



(a) Proxy Mobile IPv6 (PMIPv6)-based test-bed (b) SDN-based test-bed

Figure 5.1: Test-beds used in the experimental evaluation.

mobility solutions devises any intervention on the MN, the hardware and software requirements for the MNs are loose, being simply an IEEE 802.11b/g wireless card, and a standard IPv6 stack implementing Neighbour Discovery [Nar+07]. The following paragraphs delve into the solution-specific test-beds description.

The PMIPv6-based test-bed is depicted in Figure 5.1(a). In order to make the scenario more realistic, a transport IPv6 network composed by several IPv6 routers was added to the test-bed. These routers connect the DMM-GWs to the Control Mobility Database (CMD) and to the Correspondent Node (CN). The dashed blue lines in Figure 5.1(a) represent the logical interaction between the CMD and the DMM-GWs (these lines do not represent a dedicated path between the CMD and the DMM-GWs). The DMM-GWs and the CMD are the only nodes that run the implementation of the PMIPv6-based solution, which is publicly available as open source[11]. Such implementation replicates the signalling and operations specified in [BOG18] and briefly summarised in Chapter 4.1.1.

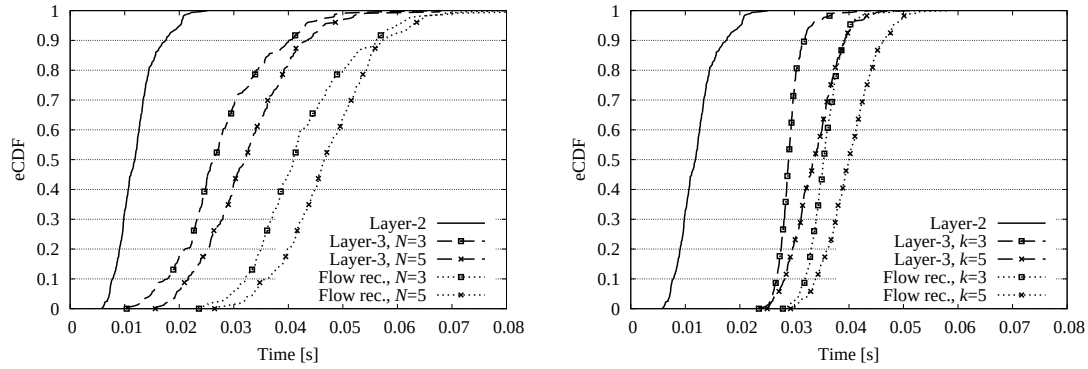
In the SDN-based test-bed, in addition to the DMM-GWs, 5 Egress Routers ER were added as illustrated in Figure 5.1(b). As it can be observed from the picture, the DMM-GWs and the Egress Router (ER)s are connected each other through two separate networks, one for the control plane (drawn with blue lines) and one for the data plane (the black solid lines). In the control plane network, a switch realises the interconnection among all the nodes and also with the Network Controller (NC) (see the CP Switch node in Figure 5.1(b)). For the data plane, the packet forwarding within the network is based on Virtual LAN (VLAN)s and statically configured. Thus, an 802.1Q-capable switch is deployed and configured in the data plane so as to interconnect all the DMM-GWs and ERs. Moreover, the ERs have a third link used to connect the test-bed to the CN.

Since the SDN-based solution uses OpenFlow as Southbound API, all the DMM-GWs and ERs run the version 3.10 of Linux kernel. This version of the kernel includes Open vSwitch[12] which provides an OpenFlow 1.3 interface. The NC runs Ryu[13] as OpenFlow controller and implements the framework proposed in Chapter 3. The SDN-based solution is therefore implemented as Ryu application (i.e., based on the API running on the NC). The connection between Open vSwitch and Ryu is performed out-of-band involving TCP for the OpenFlow messages delivery. The

[11] MAD-PMIPv6: `https://github.com/ODMM/MAD-PMIPv6`
[12] Open vSwitch: `http://openvswitch.org/`
[13] Ryu: `http://osrg.github.io/ryu/`

Table 5.3: Handover latency experimental results in milliseconds.

TYPE OF SOLUTION		LAYER-2 HO.		LAYER-3 CONF.		IP FLOW REC.	
		Mean	Std. dev.	Mean	Std. dev,	Mean	Std. dev.
PMIPv6-based	$N = 2$	12.9	4.4	25.8	8.6	40.3	9.3
	$N = 3$	12.9	4.4	27.8	8.7	42.5	9.4
	$N = 5$	12.9	4.4	32.6	8.9	47.4	9.7
SDN-based	$k = 2$	12.9	4.4	26.8	2.3	33.0	2.8
	$k = 3$	12.9	4.4	29.2	2.4	35.6	3.0
	$k = 5$	12.9	4.4	33.7	4.4	40.2	4.8



(a) PMIPv6-based solution (b) SDN-based solution

Figure 5.2: Flow-recovery time eCDF.

application is in charge of all the tasks described in Chapter 4.2. Finally, the implementation the SDN framework, including the SDN-based DMM solution, has been published as open source by the author of this thesis.[14]

5.2.1 Experimental results of PMIPv6- and SDN-based DMM solutions

For both implementations, the three handover events introduced in Chapter 4.3.3 have been measured. Wireshark[15] was used as packet sniffer, installed in the MN to measure the intervals detailed in the following:

1. *Layer-2 handover.* It is measured as the interval between two IEEE 802.11 control messages: *Deauthentication*, sent by the MN to the old DMM-GW, and *Association response* received by the MN from the new DMM-GW.
2. *Layer-3 configuration.* It is the time spent since the *Deauthentication* message, to the instant when an Router Advertisement (RA) message is received by the MN[16].
3. *IP flow recovery.* It is measured as the time required to recover *ping* traffic generated by a correspondent node to the MN every 2 ms, which is below the average the MN-CN Round Trip Time (RTT). This corresponds to the interval between the last *ping* packet received or sent by the MN before the handover and the first *ping* packet received or sent after the handover.

Figure 5.2 depicts the empirical Cumulative Distribution Function (eCDF) of the above components from the values obtained from few hundreds handovers when $N = 3,5$ in the PMIPv6-based case, and $k = 3,5$ in the SDN-based case. Table 5.3 summarises the experimental results reporting the mean and the standard deviation values also in the cases $N = 2$ and $k = 2$.

As expected, the *Layer-2 handover* does not depend on the mobility protocol. The table reports

[14] The SDN-based DMM solution is available at: `https://github.com/ODMM/openflow-dmm`
[15] `http://www.wireshark.org/`
[16] The IPv6 Duplicate Address Detection is disabled since the prefix is uniquely assigned to the MN.

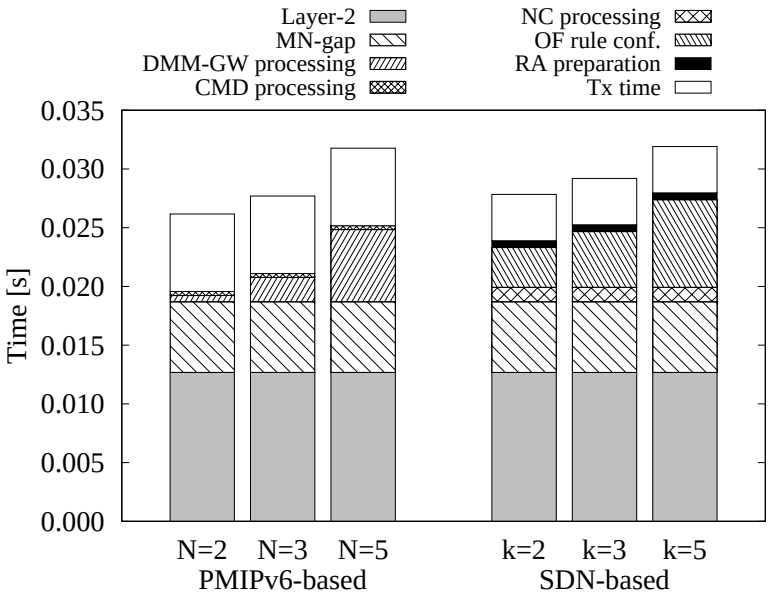


Figure 5.3: Layer-3 latency composition.

the same value for all the configurations because the measured difference was negligible in the testing system. For what concerns the *Layer-3 configuration*, i.e., how long it takes for the MN to gain IP connectivity with the DMM-GW, it can be observed that the two protocols behave similarly on average, showing a lower variance in the SDN case due to some system-level optimisation applied to the node acting as network controller. The *IP flow recovery* (i.e., *ping*) time exhibits the largest gain in favour of the SDN approach. This is due to the use of tunnelling by the PMIPv6-based solution, which was observed to introduce, on average, around 15 ms of additional delay with respect to the Layer-3 configuration, for all values of N. On the contrary, the SDN-based solution accomplishes the ping recovery with less than 7 ms of additional delay.

In order to better understand how the two solutions scale, Figure 5.3 explores in detail the components of the Layer-3 configuration time for varying values of the number of active DMM-GWs, $N = 2,3,5$ in the PMIPv6-based case, and the number of egress routers, $k = 2,3,5$ in the SDN-based case. As it can be noticed from the results, the Layer-2 switch time is the major contributing term in all set-ups More, 5 ms gap was observed, denoted as "MN gap", between the instant the MN receives the *Association response* message, and the time it sends the RS message to the DMM-GW. This gap could be removed by employing a dedicated detection mechanism for the Layer-2 link activation and de-activation. After the link-up phase, the following components were separated: (*i*) the component due to message transmission, which depends on the sum of the RTT in the radio link between the MN and the DMM-GW, and (*ii*) the RTT in the wire between the DMM-GW and the CMD or NC, respectively for the PMIPv6-based or the SDN-based solution. In the laboratory tests, all the nodes are close to each other, and such RTT sum is less than 5 ms. In a real deployment, with larger RTT values, the Layer-3 configuration time would tend to approximate the air time plus the distance from the central node to the farthest router involved in the signalling. The above components do not significantly vary with the increasing number of N and k, whereas such parameters impact the processing at the network nodes, confirming the intuition that T_P tends to grow with the number of entities involved in the handover operations. In the PMIPv6-based case, the heaviest burden is on the DMM-GW, because of the tunnels and routes set up, so, the larger is the number of previous DMM-GW, the longer is the latency. The CMD is mainly answering to a query, so its task is accomplished much quicker in approximately constant time. In the SDN based case, the NC has to compute and send the rules to the Egress Routers. In addition it has to process the Router Solicitation (RS) from the MN and prepare the RA message. From Figure 5.3, it can be observed that the variable components approximately grow linearly in both solutions.

The results reported in this Chapter show that the SDN-based DMM approach has similar or better performance figures than previous DMM protocols based on PMIPv6. This is an encouraging outcome to foster further optimisation for future deployments of such kind of solutions. For example, further evaluation is required to analyse scenarios where multiple MNs simultaneously attach to the network or perform a handover, thus producing various requests overlapping in time at the NC side. For instance, in these cases the NC is expected to be backlogged because of all the concurrent MN requests. Similarly, multiple flow rules are expected to be configured at the same time on the same SDN switch, e.g., when several MNs simultaneously perform an handover to the same target DMM-GW. These would eventually lead to an increase of processing and configuration time, thence to a higher overall handover latency experienced by the MNs. Evidently, a SDN-based DMM solution should consider also the distribution of the control plane in addition to the data plane. This implies having multiple replicas of the DMM solution modules in the network which still need to provide a harmonised mobility support to the MNs whilst providing a bound to the handover latency. Therefore, an analogous methodology as the one used in this Chapter can be then used in such scenarios to analyse the breakdown of the handover latency and to derive some deployment options for the SDN-based DMM solution, e.g. the number of replicas and where to deploy them to support a higher volume of MNs and different mobility patterns. These topics are left for future work by the authors.

The evaluation focuses on the handover latency produced by this solution, using NS-3 simulations whose results are aligned with the ones experimentally obtained in this thesis. Similar to the experimental validation conducted in this thesis, another test-bed-based validation using commodity hardware and Wi-Fi access is available in [SOM16][17]. The solution therein proposes a hierarchy of SDN controllers in order to handle intra-district handovers, i.e., within an access network handled by the same DMM-GW), and inter-district handovers, i.e., including DMM-GW relocation. Due to the scenario discussed by the solution and additional complexity required in the signalling, the results reported in [SOM16] for the handover latency appear slightly larger than those presented in the present research. Yet another SDN-based DMM architecture is validated experimentally through a set-up employing a Ryu controller and a mininet-created test network in [NBH16]. The SDN solution therein handles packet redirection after handover in two different ways, both different from the proposed solution. The first method is the tunnel mode, i.e., establishing a tunnel between the source and target access gateways in order to convey re-directed packets; the second method is based on route optimisation, that is a full path computation and the subsequent population of the forwarding rules onto the in-path switches. The numeric results show similar values for the handover latency as those obtained in this thesis, but they are not directly comparable as there is no wireless access employed in the test-bed described in [NBH16]. In addition, it is argued that the redirection methods do not scale as well as the one proposed in the present solution, especially the method employing a full path reconfiguration.

5.2.2 Comparison with existing SDN deployments with mobility support

One of the most remarkable SDN deployments applied to wireless networking is *OpenRoads* [Yap+09][18] (also known as *OpenFlow Wireless*) developed at Stanford University, open to researchers for running their algorithms concurrently by means of virtualisation. OpenRoads incorporates different wireless technologies, namely Wi-Fi and WiMAX, and one of its early proofs of concept was based on providing mobility across multiple technologies [Yap+10a][19]. In addition, the performance of

[17] M. I. Sanchez, A. de la Oliva and V. Mancuso. 'Experimental Evaluation of an SDN-based Distributed Mobility Management Solution'. In: *Proceedings of the Workshop on Mobility in the Evolving Internet Architecture*. MobiArch '16. New York City, New York: ACM, 2016, pages 31–36. ISBN: 978-1-4503-4257-5. DOI: 10.1145/2980137.2980138.

[18] K.-K. Yap et al. 'The Stanford OpenRoads Deployment'. In: WINTECH '09. Beijing, China: ACM, 2009, pages 59–66. ISBN: 978-1-60558-740-0. DOI: 10.1145/1614293.1614304.

[19] K.-K. Yap et al. 'Blueprint for Introducing Innovation into Wireless Mobile Networks'. In: *Proceedings of the Second ACM SIGCOMM Workshop on virtualized Infrastructure Systems and Architectures*. VISA '10. New Delhi, India:

OpenRoads has been demonstrated by means of an n-casting transmission solution [Yap+10b][20]. All these approaches are based on the same principle as the proposed mobility approach, reconfiguring the data-path, although they do not consider IP mobility or the design of a scalable architecture as we do. All the tools used by OpenRoads are open source, so as to make the infrastructure reproducible by other research groups in their own networks. Likewise, the presented implementation shows the flexibility of current SDN software tools available as open source and is built upon commercial-off-the-shelf devices.

A full-fledged software-defined mobile network (SDMN) is defined in MobileFlow [PWH13][21] and its authors provide a comparison to the current Evolved Packet Core (EPC) architecture. Although no numeric results are reported, a prototype implementation is also proposed in [PWH13]. The purpose is to show a proof of concept of the MobileFlow Forwarding Engine (MFFE), which encompasses all the user plane protocols and functions, and the MobileFlow Controller (MFC), which is a logically centralised entity that configures dynamically the MFFEs (i.e., the data plane). Despite MobileFlow is OpenFlow-based, MFFEs must also support operations that are not carried out at the switch-level, as layer-3 tunnelling, for instance. Mobility management can be supported as the controller can update forwarding rules according to the tunnel encapsulation or de-capsulation requirements. This approach is also followed in the implementation, where we can set the tunnelling and forwarding rules above link layer and the controller updates the forwarding rules in the OpenFlow domain. A different approach presented in [Kem+12][22] proposes to move the EPC to the cloud by means of virtualisation and implementing GPRS Tunnelling Protocol (GTP) extensions for OpenFlow for mobility management. The mobility solutions proposed by these works are not really inspired by the DMM paradigm, but are rather based on the same mobility concepts currently used in cellular networks, hence inheriting the scalability issues of traditional mobility approaches such as PMIP or GTP-based mobility management.

Regarding mobility management solutions relaying on the SDN paradigm for session continuity management, in [WB14][23] the authors propose a solution based on IP translation across the mobility domain. This solution differs mainly with ours on the mobile terminal support. While the proposed mobility solution is completely transparent for the mobile terminal, the solution presented in [WB14] requires the terminal to bind its current location to its identifier, using Mobile IP signalling. The solution is supported by a mininet[24]-based proof of concept, whose authors use to observe the behaviour of TCP flows during handovers handled by their solution, compared to plain PMIPv6. A similar approach can also be found in [Li+13][25]. This work uses a SDN-based approach for path modification, but it also provides of extensions to OpenFlow to update the Domain Name System (DNS) of the network with the new location of the user. The use of this extensions highly impact on the handover performance which is heavily increased with respect to the solution proposed in this thesis. Similar approaches as the one defined in this thesis above can also be found in the literature. For example, authors in [KVK14][26] propose a system using

ACM, 2010, pages 25–32. ISBN: 978-1-4503-0199-2. DOI: 10.1145/1851399.1851404.

[20] K.-K. Yap et al. 'OpenRoads: Empowering Research in Mobile Networks'. In: *SIGCOMM Comput. Commun. Rev.* 40.1 (January 2010), pages 125–126. ISSN: 0146-4833. DOI: 10.1145/1672308.1672331.

[21] K. Pentikousis, Y. Wang and W. Hu. 'Mobileflow: Toward software-defined mobile networks'. In: *IEEE Communications Magazine* 51.7 (July 2013), pages 44–53. ISSN: 0163-6804.

[22] J. Kempf et al. 'Moving the mobile Evolved Packet Core to the cloud'. In: *2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. October 2012, pages 784–791. DOI: 10.1109/WiMOB.2012.6379165.

[23] Y. Wang and J. Bi. 'A solution for IP mobility support in software defined networks'. In: *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*. August 2014, pages 1–8. DOI: 10.1109/ICCCN.2014.6911783.

[24] http://mininet.org

[25] Y. Li et al. 'Software defined networking for distributed mobility management'. In: *2013 IEEE Globecom Workshops (GC Wkshps)*. December 2013, pages 885–889. DOI: 10.1109/GLOCOMW.2013.6825101.

[26] M. Karimzadeh, L. Valtulina and G. Karagiannis. 'Applying SDN/OpenFlow in Virtualized LTE to support Distributed Mobility Management (DMM)'. in: *Proceedings of the 4th International Conference on Cloud Computing and Services Science (CLOSER 2014)*. SCITEPRESS, April 2014, pages 639–644. ISBN: 978-989-758-019-2. DOI:

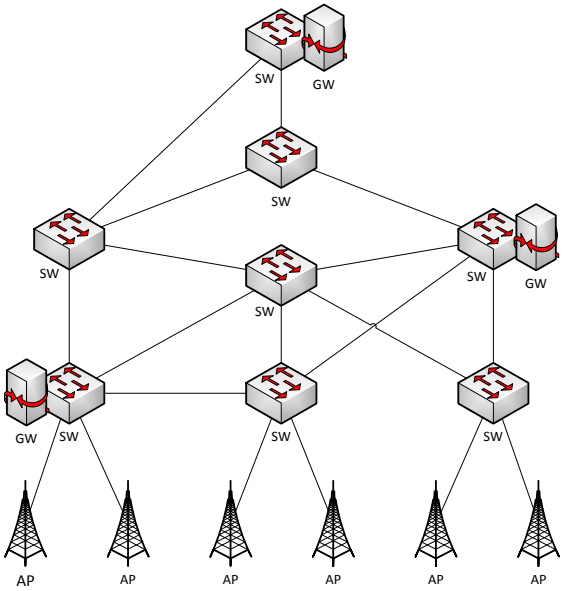


Figure 5.4: Test-bed topology.

Table 5.4: UE handover time results (in milliseconds) for different handover rate λ at the network controller.

	$\lambda = 6$			$\lambda = 12$			$\lambda = 30$			$\lambda = 60$		
	L2	L3	Ping	L2	L3	Ping	L2	L3	Ping	L2	L3	Ping
Mean	45	104	113	45	104	113	045	110	120	45	136	151
Standard deviation	29	32	32	29	32	32	29	54	54	29	75	94
95^{th} percentile	125	184	193	125	186	193	125	206	215	125	264	336

OpenFlow to manage the mobility of users in 3rd Generation Partnership Project (3GPP) networks. Differing from the proposed SDN-based DMM solution, this work is a conceptual analysis on how SDN can be applied to 3GPP networks and does not include any implementation or empirical evaluation. Nevertheless, the same authors further elaborated their solution, leading to [Val+14][27]. The article proposes an SDN-based DMM approach for virtualised Long Term Evolution (LTE) systems which leverages on an interface between the SDN controller and the MME, in order to detect an IP anchor change (i.e., a Packet Data Network Gateway (PGW) relocation). When such a relocation occurs, the SDN controller re-routes ongoing traffic to the new PGW.

5.3 Evaluation of SDN controller scalability

With the purpose of evaluating the performance of the NC implementation, the network topology shown in Figure 5.4 has been configured on the SDN test-bed. The test-bed comprehends 14 switches and a network controller interconnected through Ethernet (IEEE 802.3). Moreover, 3 switches expose IP gateway capabilities and 6 switches offer IEEE 802.11b/g connectivity to the MNs. As the SDN architecture does not devise any intervention on the MN, its hardware and software requirements are simply an IEEE 802.11b/g interface, and a standard IP stack. The network topology is based on the results and trends reported in [Ber+14][28] and it is composed of three

10.5220/0004946106390644.

[27] L. Valtulina et al. 'Performance evaluation of a SDN/OpenFlow-based Distributed Mobility Management (DMM) approach in virtualized LTE systems'. In: *2014 IEEE Globecom Workshops (GC Wkshps)*. December 2014, pages 18–23. DOI: 10.1109/GLOCOMW.2014.7063379.

[28] C. J. Bernardos et al. 'An architecture for software defined wireless networking'. In: *IEEE Wireless Communications* 21.3 (June 2014), pages 52–61. ISSN: 1536-1284. DOI: 10.1109/MWC.2014.6845049.

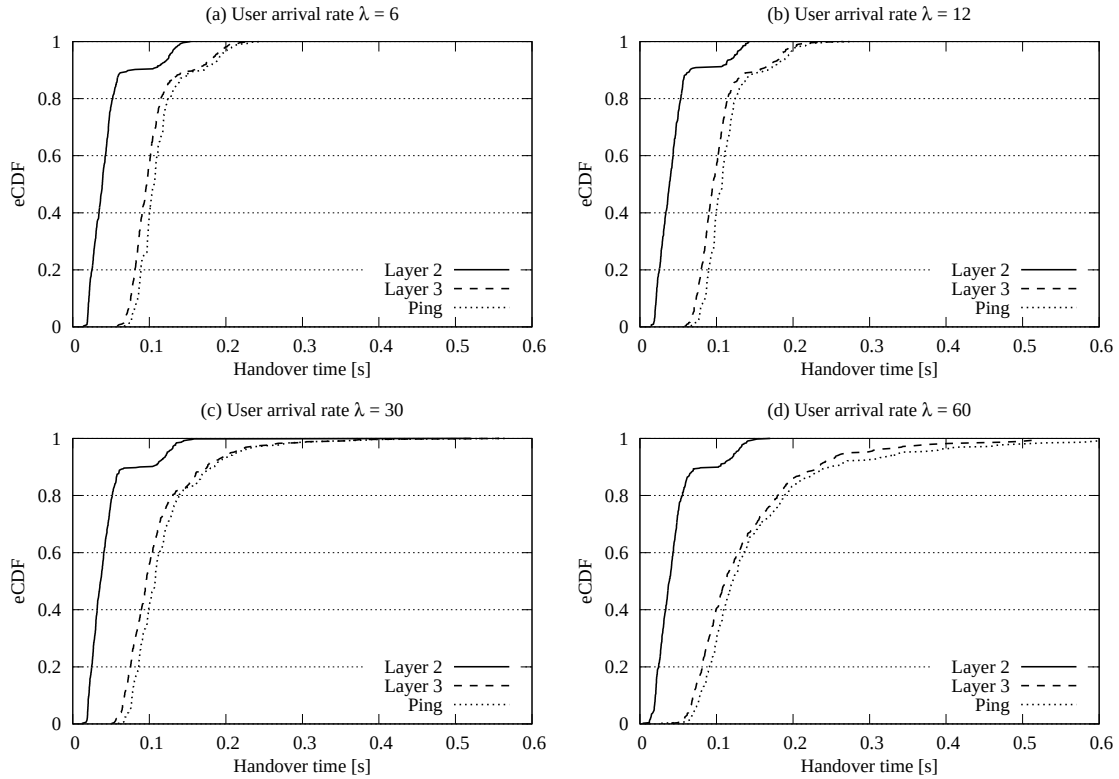


Figure 5.5: Flow-recovery time eCDF with different λ values.

layers: (*i*) Radio Access Network (RAN), (*ii*) aggregation network, and (*iii*) core transport network. The whole network is managed by a single controller but in a wider scenario multiple controllers might be employed (i.e., one per each network layer). In the same way, in this implementation all the MNs are managed by the same controller. Clearly, a single controller cannot process all the handover/attachment requests in a real scenario where millions of MNs are simultaneously connected to the network. Therefore, multiple controllers will be employed and each controller will be in charge of a separate set of MNs, e.g., a controller is in charge of all the MNs in a specific geographical area. Nonetheless, it's still unclear how many MN requests can be effectively managed by a single controller. Clearly, the absolute numbers reported in the following are related to this implementation and more requests could be supported by employing a more powerful server for the network controller. Nevertheless, every software component running in the NC is evaluated and dissected in order to provide a complete insight of signalling management. Particularly, the handover delay (i.e., the time an MN does not have connectivity as a result of a change of point of attachment) is analysed by identifying the components that affect more the overall latency. For this analysis, a node external to the test-bed generates ping traffic destined to the MN every 1 ms. The handover delay analysis can be analysed looking at three different aspects: (*i*) the Layer 2 handover: time elapsed since the old radio link is torn down until the new one is established; (*ii*) the Layer 3 configuration: time required by the MN to obtain network layer connectivity (including the Layer 2 handover); and (*iii*) the IP flow recovery: time interval during which an IP flow is interrupted due to the handover (including both the Link 2 and the Layer 3 configuration). The main component of Layer 3 configuration time is given by the processing at the network controller.

With the goal to evaluate the scalability of the network controller, 100 simultaneously attached users perform a handover following a Poisson process with different rate λ. Specifically, one physical MN performs an handover while other 100 MNs are emulated. From a NC perspective, the emulated MNs are treated in the same way of the physical way, involving all the mobility procedures and data plane configuration. The results reported in the following are for the physical MN, which are affected at controller side by the 100 emulated MNs. Therefore, the Layer 2

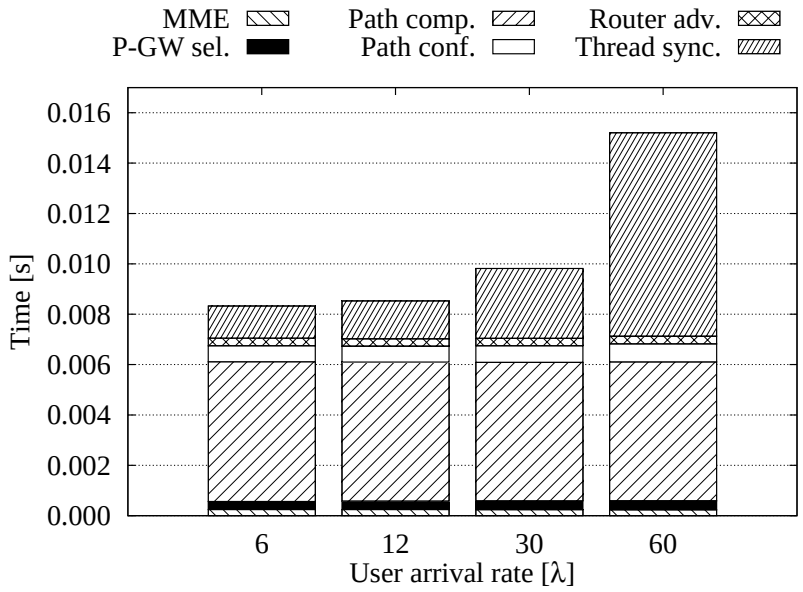


Figure 5.6: Processing time for different handover rates λ at the network controller.

handover time of the physical MN is not affected by any other MN contending the same radio channel. Figure 5.5 depicts the eCDF of the handover time for different λ as experienced by the MN, while Table 5.4 reports the numerical values of obtained results with the most significant statistical properties. As it can be noticed, the Layer 2 is independent of λ, on the contrary the time required to obtain Layer 3 configuration increases for higher λ values. This is due to the higher number of operations run in the network controller because of greater handover rate. Additional tests were performed with the purpose of identifying the main components of the increased processed time as explained in the following. The network controller relies on a multi-thread implementation in order to take advantage of modern multi-core CPUs whereas each thread accomplishes a specific task. In particular, one thread manages the connection with the MME, a second thread is in charge to select the best PGW for the MN upon an handover, while a third thread takes care of communicating the IP parameters to the MN (i.e., sending a RA for IPv6 connections). Two additional threads are in charge of computing and configuring the paths within the network.

Figure 5.6 depicts the processing and synchronisation time for each thread for different λ. As it can be noticed, the processing time is independent of the rate λ. This is due to the fact that, according to this implementation, the operations required for managing a single handover do not depend on the numbers of handovers occurring simultaneously. On the contrary, the synchronisation time increases with λ. A higher synchronisation time means that the message queue system is getting more and more congested. Two potential causes of such behaviour are the following: (*i*) the message queue system does not scale adequately or, (*ii*) the processing time is greater than the handover rate. Our tests showed a CPU occupation growth according to λ, in particular, the CPU was 100% busy for $\lambda = 60$. This means that the network controller is always backlogged and it is not able to serve all the handover requests in time. Indeed, this occurs when the processing is not fast enough compared to the arrival rate leading to an increase in the overall handover time experienced by the MNs. In order to overcome the problem, distribution of the network controller functionality could be proposed for scaling up, following some of the existing techniques for high-performance distributed systems [Cou+11][29].

[29] G. Coulouris et al. *Distributed Systems: Concepts and Design*. 5th. Addison-Wesley Publishing Company, 2011. ISBN: 978-0132143011.

6. Conclusions of Part One

Software Defined Networking (SDN) is seen as one of the key tool to provide *enhanced flexibility* in future network architectures. To that end, the first part of this thesis departed from the general SDN framework defined by the Open Networking Foundation (ONF) and fully designed a compatible architecture suitable for future network operators. Among the plethora of use cases and foreseen services, Distributed Mobility Management (DMM) is considered a necessity in future mobile network deployments in order to offload the network core from traffic that can be locally routed close to the access. Different actors have been working on this area, being the Internet Engineering Task Force (IETF) a major venue where most of the solutions have been discussed so far, while 3rd Generation Partnership Project (3GPP) has more recently started to work on distributed mobility architectures. Although there have been many different proposals, most of them share a characteristic: they are an evolved version of current IP mobility based solutions. While these are enough to offload the network core, and pose no significant deployment concerns, operators are already looking into SDN-based DMM solutions since they can potentially reduce the complexity and costs incurred by service creation and network operation.

> Publications covering the design of the SDN framework for quick service provisioning, including related concepts, are [L C+18b], [Wan+15a].

To that end, this part analysed how an SDN-based solution might look like when providing DMM support. Specifically, it presented the analytic and experimental evaluation of two key DMM protocol families: IP mobility and SDN-based. While the Proxy Mobile IPv6 (PMIPv6)-based solution is available in literature, this part thoroughly designed, modelled, and implemented the SDN-based solution. Additionally, this part walked the path of decomposing the functions that a DMM solution should have and identify how these can be implemented in an DMM-based solution. Moreover, existing state-of-the-art solutions are not generally studied both analytically and experimentally as it is done in this part, thus providing solid insights on how to apply DMM concepts in future mobile networks. By implementing the proposed SDN architecture and testing it on a medium size test-bed, this part demonstrated how easy and quick would be for an operator to create and put into operation new services, like the proposed SDN-based DMM. The results obtained from analysis and experiments show that the performance of the analysed solutions depends on the scenario being considered, but also indicate that SDN approaches have a big potential: (*i*) achievable performance is good and even better than the one of the PMIPv6-based solution, (*ii*) the solution can be easily implemented, and (*iii*) provides additional flexibility in regards of how it behaves and provides service differentiation.

Publications covering the design of the design and the experimental assessment of the SDN-based DMM contributions, including related concepts, are [Con+16], [GLB15], [L C+17a].

An open source SDN-based DMM implementation called OpenFlow-DMM is available at `http://odmm.net/openflow/`.

While this part focused on the SDN control plane (and the flexibility achievable via programming the Network Controller (NC)), a fundamental next step is to analyse the SDN data plane so as to ensure that it provides the necessary primitives for a full programmability of the network. To that end, and in order to unleash the full potential of SDN, the underlying network needs to undergo a profound transformation, evolving from a static connectivity substrate towards a service-oriented infrastructure capable of accommodating the various 5G services, including high-bandwidth and latency-sensitive scenarios. The next part of this thesis therefore focuses on this data plane aspect of the SDN paradigm.

II Part Two

7. Designing a SDN-based 5G transport network

The *backhaul* is a network segment that comprises the links between the core network and access network. It provides the connection of the local cell site through the core network (e.g., gateways) to the application servers (e.g., Internet). A backhaul network is composed of several sub-systems including (*i*) heterogeneous wired/wireless forwarding networks typically with tree and chain topologies towards the access (cell sites), and (*ii*) fibre-based aggregation and routing networks typically in ring and mesh topologies towards the core network (e.g., controllers/gateways). The backhaul network infrastructure includes several network nodes, such as switches, bridges, routers, aggregators, etc., and it uses various transport protocols to communicate between these nodes such as Ethernet, carrier-grade Ethernet, Optical Transport Network (OTN), Synchronous Digital Hierarchy (SDH), Multiprotocol Label Switching (MPLS), IP, etc. The heterogeneity of the backhaul network infrastructure, both in terms of the various physical and logical connections, makes it quite complex to control and manage flexibly. Recently, a new network segment called *fronthaul* has emerged, as the result of more Centralised RAN (C-RAN) architectures aimed mainly at driving down the operators total cost of ownership whilst enabling better performance for new schemes, such as Coordinated Multipoint (CoMP) through joint processing across multiple base stations (e.g., Evolved Node B (eNB) in 4G, Next Generation NodeB (gNB) in 5G). In C-RAN architectures, the base station is split into two elements, a Radio Unit (RU) and a Digital Unit (DU). In 4G, the RU is also referred to as Remote Radio Head (RRH) because it simply keeps the RF functions necessary for the signal radiation at the cell site, while the DU is also referred to as Baseband processing Unit (BBU) because it takes all the baseband heavy computational functions to a central location (e.g., central office or cloud). In 5G, the static split of functionality evolves towards multiple functional splits in order to enable the dynamic configuration and separation of the gNB functionalities between the RU and the DU.

Nowadays, the fronthaul and backhaul segments are two physically separated networks encompassing several levels of interconnection and transport technologies. In order to meet the 5G requirements in a cost-effective manner, a unified 5G transport network that unifies the data, control, and management planes is hence required. Such an integrated fronthaul/backhaul transport network, denoted as crosshaul, needs to carry both fronthaul and backhaul traffic operating over heterogeneous data plane technologies, which are software-controlled so as to adapt to the fluctuating capacity demand of the 5G air interfaces. To make the 5G transport network as independent as possible on the specific radio implementation, the crosshaul transport network should also support any kind of splitting option of the 5G radio protocols between remote and processing sites. Similarly, the crosshaul transport network should also support any type of 5G services.

7.1 Analysis of crosshaul requirements and transport technologies

A necessary step for the definition of the crosshaul network is hence the identification of the traffic requirements, especially in terms of latency and bandwidth, as well as a reference architecture and the most suitable transport technologies and multiplexing strategies.

7.1.1 Fronthaul and backhaul traffic requirements

Fronthaul interfaces pose tight requirements in terms of required bandwidth and admissible latency and jitter. Specifically, the closer the functional split to the RF layer, the tighter the requirements become. That is why the requirements for the 4G Common Public Radio Interface (CPRI) interface are quite tight, which makes this interface quite rigid (static) and costly. For example, a single-sector Long Term Evolution (LTE) 2x2 MIMO base station with a 20 MHz bandwidth channel requires a maximum backhaul throughput of 150 Mbps, with an average of 21 Mbps, while the CPRI fronthaul requires a constant capacity of 2.457 Gbps for an end-user maximum data rate of 150 Mbps, according to the NGMN Alliance [NGM15b][1]. In addition to this, CPRI demands about 100 μs of maximum end-to-end latency for up to 10 kms of distance between RRH and BBU [IEE15a][2] with a strict link delay accuracy of $\pm$8 ns difference between master clock and slave ports, and 2 ppb frequency deviation from the CPRI link to the BBU. Therefore, a packet-switched network requires a jitter compensation buffer and a careful Quality of Service (QoS) engineering design in order to match the CPRI delay accuracy requirements. For instance, 50 μs of latency budget are spent on propagation over 10 kms of fibre and the remainder 50 μs are the margin for switching [Oli+16][3].

Compared to LTE/LTE-A, 5G should achieve a hundredfold data rate growth (10 Gbps) for each sector. Such data rate requires an ultra-high fronthaul capacity with a very low latency. In such context, the best transmission medium to meet LTE and 5G requirements is fibre. However, a fibre deployment may not be cost-effective in some cases. This is why different functional splits have been defined on the attempt to relax the requirements of today's fronthaul and reach a more scalable interface for the future, so that lower cost and flexible means to transport the fronthaul traffic can be used. In addition, 5G Infrastructure Public Private Partnership (5G-PPP) defines a broader set of Key Performance Indicator (KPI) for future 5G networks [5GP14][4]. 1000 times higher wireless area capacity and more varied service capabilities shall be provided with respect to that offered in 2010. Very dense deployments of wireless communication links to connect over 7 trillion wireless devices serving over 7 billion people shall be facilitated. Similarly, the next-generation wireless technologies shall allow ubiquitous connectivity also in low density areas, a necessary requirement for Internet of Things (IoT) and smart cities scenarios.

To tackle the wide variety of scenarios foreseen in 5G, the 3rd Generation Partnership Project (3GPP) has defined a set of services with the corresponding requirements in [3GP18i], [3GP18j]. Those services are grouped in three categories, namely network slices: (*i*) enhanced Mobile Broadband (eMBB), (*ii*) Ultra-Reliable and Low Latency Communications (URLLC), and (*iii*) Massive Internet of Things (MIoT). eMBB services are characterised by high *bandwidth* requirements, spanning from few Mbps to 1 Gbps per user, and by moderate-latency requirements, with the most stringent one being 2-4 ms for virtual meetings. Instead, URLLC services are characterised by low *latency* and high-reliability requirements. Tactile interaction and remote motion control for

[1] NGMN. *Backhaul and Fronthaul evolution*. White Paper v1.01. Next Generation Mobile Networks Alliance, March 2015.

[2] IEEE. *CPRI requirements for Ethernet Fronthaul*. Working Group discussion 802.1CM. Institute of Electrical and Electronics Engineers (IEEE), November 2015. URL: http://www.ieee802.org/1/files/public/docs2015/cm-CPRI-requirements-1115-v01.pdf (Retrieved: 10th January 2019).

[3] A. de la Oliva et al. 'An overview of the CPRI specification and its application to C-RAN-based LTE scenarios'. In: *IEEE Communications Magazine* 54.2 (February 2016), pages 152–159. ISSN: 0163-6804. DOI: 10.1109/MCOM.2016.7402275.

[4] 5G-PPP. *5G PPP Key Performance Indicators*. 5G Infrastructure Public Private Partnership, 2014. URL: https://5g-ppp.eu/kpis/ (Retrieved: 10th January 2019).

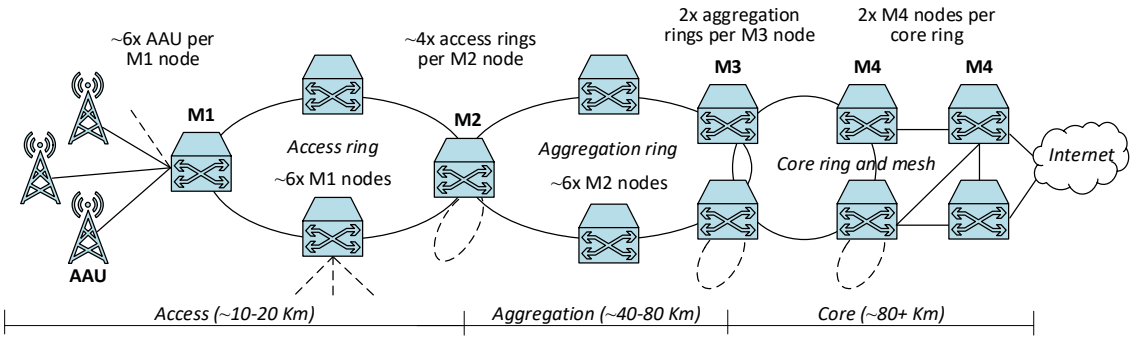


Figure 7.1: Reference 5G crosshaul network architecture based on [ITU18b].

robots require a maximum end-to-end delay of 0.5-1 ms with a *jitter* of 100 μs. Moreover, URLLC defines a survival time[5] for the services, ranging from 10 to 100 ms, with a service availability up to 99.9999%. Finally, MIoT metrics relate more to the capability of the network system to handle millions of active connections generating sporadic traffic.

7.1.2 Reference crosshaul network architecture

In order to fulfil the fronthaul and backhaul traffic requirements, the transport network architecture has to also go through some transformations as compared to current deployments. Figure 7.1 illustrates the 5G transport network reference architecture as recently proposed by the ITU Telecommunication Standardization Sector (ITU-T) in [ITU18b][6]. The transport architecture comprises three segments: (*i*) access, (*ii*) aggregation, and (*iii*) core. The access comprises on average 6 Active Antenna Unit (AAU)[7] for each node M1 connected via a point-to-point link, and ∼6 M1 nodes connected in a ring topology. Thus, each access ring connects a total of 36 AAU on average. Next, each aggregation ring comprises ∼6 M2 nodes, each of which serves as gateway to 4 access rings on average. Each aggregation ring is served by two M3 nodes for redundancy reasons, while each M3 node provides gateway capabilities to 2 aggregation rings. It is worth noticing that the M1 and M2 nodes are configured in a ring topology (i.e., access and aggregation rings, respectively) only at electrical level while at logical level are considered to be connected point-to-point to their corresponding gateways (i.e., M2 and M3, respectively). This means that packets are enqueued only at gateway level and not every time they traverse a node in the ring. Next, the mobile packet core network comprises two M4 nodes for each core ring and a variable number of other M4 nodes connected in a mesh fashion. The amount of M4 nodes highly depends on the physical deployment of the mobile network at country level. For the sake of example, [Nau+][8] reports 12 nodes M4 in the case of Germany.

7.1.3 Transport technologies for a crosshaul network

Given the wide variety of deployment scenarios envisioned in 5G, it is unthinkable to adopt a one-fits-for-all approach for designing the crosshaul transport. Therefore, the design of the 5G data plane requires a holistic view, encompassing radio protocols, optical and wireless transmission, packet switching, and integrated and intelligent management. To that end, a crosshaul network for 5G scenarios is expected to combine different transport technologies in order to create a unified data plane for crosshaul traffic. Specifically, the crosshaul data plane architecture envisages an

[5] The survival time is the time that an application consuming a communication service may continue to operate without receiving any messages.

[6] ITU-T. *Consideration on 5G transport network reference architecture and bandwidth requirements*. Study Group 15 Contribution 0462. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), February 2018.

[7] The term Active Antenna Unit (AAU) in ITU-T corresponds to the term Radio Unit (RU) in 3GPP.

[8] B. Naudts et al. 'How can a mobile service provider reduce costs with software-defined networking?' In: *International Journal of Network Management* 26.1 (), pages 56–72. DOI: 10.1002/nem.1919.

inter-connected packet-circuit switched transport network so as to combine bandwidth efficiency with deterministic latency. The packet switched network is the primary path for the transport of most delay-tolerant fronthaul and backhaul traffic, whereas the circuit switched network (mainly optical) complements the packet switched path for those particular traffic profiles that are not suited for packet-based transport (e.g., legacy CPRI or traffic with extremely low delay tolerance).

Wireless networks

Wireless links are a suitable solution at the edge of the mobile transport network, to increase capillarity, when optical fibre or copper connections are not available or economically viable. However, traditional frequencies below 50 GHz are already very crowded and fragmented. The current focus of industry is moving towards higher frequency bands, from 50 to 90 GHz (i.e., mmWave), where large contiguous spectrum chunks exist. Regarding backhaul networks, a reconfigurable mesh mmWave point-to-multipoint (PtMP) backhaul network would enable the deployment and cost advantages of current sub-6 GHz PtMP non-line-of-sight (NLoS) systems while offering higher capacity (e.g., tens of Mbps per user). In the case of fronthaul, novel spectrally efficient modulation techniques [Bla+12][9], [Dat+14][10] are proposed to solve the limitations observed in current microwave and mmWave fronthaul systems in terms of rates and distance. Finally, optical wireless technology is also considered, either LED or laser-based, as an attractive alternative technology due to its unlicensed spectrum band, immunity to electromagnetic interference and performance. The LED backhaul link is able to transmit up to 500 Mbps over 100 m [Sch+16][11], while laser based devices provide bit rates up to 10 Gbps over a few kilometres. In addition, optical wireless can be used in conjunction with the above wireless technologies, for improving the link availability, or as gap filler in fibre links.

Fixed access networks

The reuse of current installed fibre and copper infrastructure are an appealing alternative for fronthaul or backhaul applications with respect to new deployments. A Passive Optical Network (PON) deployed in a point-to-multipoint fashion is a popular architecture for supporting fibre-based broadband access to a set of customers [ITU08][12]. PONs have several advantages, as they are very simple, easily scalable and do not need excessive maintenance, providing a cost-effective way for the overlay of mobile traffic in the context of crosshaul networks. Such a converged architecture is challenging since it has to fulfil the aforementioned requirements for the new fronthaul/backhaul services. Although Gigabit-capable PON networks offered bandwidth (2.5/1.25 Gbit/s DL/UL) may be sufficient for backhauling residential users, it is clearly insufficient for the transport of fronthaul traffic in C-RAN scenarios according to the CPRI specification [CPR15]. The advent of 10 Gbps line rate and Time and Wavelength Division Multiplexing (TWDM) [ITU15b][13] opens the door to support current and future fixed and mobile operation on the same optical infrastructure. Copper infrastructure is also abundant: the interest for crosshaul networks is mostly on Ethernet-based

[9] S. Blanc et al. *High Capacity Wireless Communications Systems and Methods*. Patent application US20130294541. E-Blink, May 2012. URL: `https://patents.google.com/patent/US20130294541` (Retrieved: 10th January 2019).

[10] P. T. Dat et al. 'High-Capacity Wireless Backhaul Network Using Seamless Convergence of Radio-over-Fiber and 90-GHz Millimeter-Wave'. In: *Journal of Lightwave Technology* 32.20 (October 2014), pages 3910–3923. ISSN: 0733-8724. DOI: `10.1109/JLT.2014.2315800`.

[11] D. Schulz et al. 'Robust Optical Wireless Link for the Backhaul and Fronthaul of Small Radio Cells'. In: *Journal of Lightwave Technology* 34.6 (March 2016), pages 1523–1532. ISSN: 0733-8724. DOI: `10.1109/JLT.2016.2523801`.

[12] ITU-T. *Gigabit-Capable Passive Optical Networks (G-PON): Physical Media Dependent (PMD) Layer Specification*. Series G: Transmission Systems and Media, Digital Systems and Networks G.984.2. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), March 2008.

[13] ITU-T. *40-Gigabit-capable passive optical networks (NG-PON2): Definitions, abbreviations and acronyms*. Series G: Transmission Systems and Media, Digital Systems and Networks G.989.1. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), October 2015.

transmission over copper cables, since they provide high bandwidth (e.g., 40 Gbps [IEE16a][14]) and allows power savings during periods with low traffic, especially interesting for backhaul purposes.

Optical networks

In absence of legacy infrastructure, optical transmission technologies are very suitable due to high achievable capacity and transmission distance. Coarse WDM (CWDM) and Dense WDM (DWDM) are examples of optical transport technologies [ITU09][15], [NW13][16]. In some particular case, like extending base station coverage inside train tunnels, analogue Radio over Fiber (RoF) is an interesting alternative to digital transmission to reduce bandwidth and latency in fronthaul links while increasing their energy efficiency. Analogue RoF just requires electrical-to-optical conversion and RF circuits, which may lead to cost saving compared to digital transmission. The work [Ye+15][17] shows and experimental C-RAN deployment with a fibre link of 20 kms using analogue RoF and PtMP PON in combination with DWDM to achieve high aggregate capacity. DWDM metro networks support multi-channel transmission over optical fibre with bit rates spanning from 1 to 100 Gbps per optical channel, but higher rate transmission interfaces are available: 400 Gbps units started to be installed in 2018, followed by 1 Tbps channels expected in 2020. Further increases will be possible with the introduction of 1 Tbps channels by means of advanced spectral compression techniques, such as Time-Frequency Packing [Sec+15][18], overcoming the Nyquist bandwidth limit that holds for digital transmission systems with orthogonal signalling. In the extreme case of Tbps transmission over both C (1530-15665 nm) and L (1565-1625 nm) bands, the aggregate capacity can be as high as 67.2 Tbps over a single optical fibre. The high aggregate capacity makes DWDM especially suitable to support broadband services and the densely populated scenarios that 5G has to support.

7.1.4 Multiplexing strategies for a crosshaul network

Multiplexing is a method by which multiple data streams are combined into one signal over a shared medium. This allows reducing the costs of deploying and operating a network. The following three multiplexing strategies are considered for a unified fronthaul and backhaul transport.

Physical layer multiplexing

The most trivial solution for C-RAN is to provide point-to-point (PtP) fibre links from the RRH to the BBU: typically, fibre access cables include from 12 to 144 fibres, making possible to upgrade an existing backhaul network to C-RAN exploiting unused fibres. Coarse and dense wavelength division multiplexing (CWDM/DWDM) are multiplexing strategies for fibre optic networks, where different types of traffic (e.g., fronthaul and backhaul) are segregated to different wavelengths. Passive CWDM provides up to 18 channels with a channel bit rate up to 10 Gbps and it is the most cost-effective deployment option since it requires neither active components nor synchronisation features. Sub-channel CWDM allows increasing channels up to 54 based on technology approaching DWDM. When the number of wavelengths increases, a colourless approach is required to simplify

[14] IEEE. *Physical Layer and Management Parameters for 25 Gb/s and 40 Gb/s Operation, Types 25GBASE-T and 40GBASE-T*. Standard for Ethernet 802.3bq. Institute of Electrical and Electronics Engineers (IEEE), June 2016.

[15] ITU-T. *Multichannel DWDM applications with single-channel optical interfaces*. Series G: Transmission Systems and Media, Digital Systems and Networks G.698.1. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), November 2009.

[16] D. Novak and R. Waterhouse. 'Advanced radio over fiber network technologies'. In: *Opt. Express* 21.19 (September 2013), pages 23001–23006. DOI: 10.1364/OE.21.023001.

[17] C. Ye et al. 'A First Demonstration of a PON-based Analog Fronthaul Solution Supporting 120 20MHz LTE-A Signals for Future Het-Net Radio Access'. In: *Asia Communications and Photonics Conference 2015*. Optical Society of America, 2015, ASu3E.2. DOI: 10.1364/ACPC.2015.ASu3E.2.

[18] M. Secondini et al. 'Optical Time–Frequency Packing: Principles, Design, Implementation, and Experimental Demonstration'. In: *Journal of Lightwave Technology* 33.17 (September 2015), pages 3558–3570. ISSN: 0733-8724. DOI: 10.1109/JLT.2015.2443876.

operation. PtP DWDM PON [ITU15b] and G.metro [ITU16b][19] propose this approach based on a pilot tone channel to control the wavelength and allowing a transparent fronthaul transmission (i.e., with no framing). Research and industry are active in studying new cost effective solutions based on integrated photonics, for instance, silicon photonics.

Time division multiplexing

A further multiplexing level can be realised mixing fronthaul and backhaul traffic on the same wavelength channel. ITU G.709 [ITU16a][20], also known as OTN, and G.989/NG-PON2 [ITU15b] are optical transport network standards that define a common framing for transporting data payloads widely adopted in Wavelength Division Multiplexing (WDM) and access networks. As of today, the practical use of CPRI over OTN and TWDM appears to be limited to the case of synchronous mapping of CPRI signals belonging to a single synchronisation island, which is in contrast with the crosshaul concept of fronthaul and backhaul integration. To overcome this issue, the crosshaul network should offer a less time-sensitive circuit multiplexing, where the multiplexed frame is synchronous to the fronthaul client signal in order to avoid degradation of synchronisation accuracy from client to line. The clock signal extracted from the fronthaul link is used by the transmitter to perform several tasks: serial to parallel conversion, removal of redundant bits, multiplexing of backhaul and fronthaul, buffering for the compensation of the difference between upstream and downstream delays, and scrambling.

Packet-based multiplexing

Packet-based multiplexing especially makes sense in the presence of multiple sources with load-dependent data rate, since this enables statistical multiplexing gain. Such load dependency is inherent for backhaul, but traditional fronthaul (e.g., CPRI) has fixed bit rate independent of traffic load. In order to multiplex backhaul and fronthaul on the same physical link, new link-level features and enhancements are needed to the Ethernet standard to support a more deterministic timing. The Time-Sensitive Networking (TSN) Task Group in Institute of Electrical and Electronics Engineers (IEEE) 802.1 [IEE][21] is developing a set of standards addressing transmission of time-sensitive data over Ethernet, with very low latency and high availability. Particularly, the IEEE 802.1CM [IEE18e][22] is defining a standard network profile for fronthaul traffic. IEEE 1914.3 addresses instead Radio over Ethernet (RoE) encapsulation enabling the transfer of In-phase and Quadrature (IQ) user-plane data [IEE18b][23], vendor-specific data and control and management information channels across an Ethernet-based packet-switched network. The IEEE 1914.1 [IEE16b] group has gone one step beyond the 1914.3 standard by addressing new functional splits compared to conventional CPRI.

7.2 Delineating the main components of a crosshaul network

This section presents the design of the two main components that enable the realisation of a crosshaul network: (*i*) the Crosshaul Forwarding Element (XFE), and (*ii*) the Crosshaul Common Frame (XCF).

[19] ITU-T. *Multichannel bi-directional DWDM applications with port agnostic single-channel optical interfaces*. Draft new Recommendation ITU-T G.metro SG 15 v0.5. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), January 2016.

[20] ITU-T. *Interfaces for the optical transport network*. Series G: Transmission Systems and Media, Digital Systems and Networks G.709. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), June 2016.

[21] IEEE. *Time-Sensitive Networking Task Group*. IEEE 802.1 Working Group. Institute of Electrical and Electronics Engineers (IEEE).

[22] IEEE. *Time-Sensitive Networking for Fronthaul*. Standards for Local and metropolitan area networks 802.1CM. Institute of Electrical and Electronics Engineers (IEEE), March 2018.

[23] IEEE. *Draft Standard for Radio Over Ethernet Encapsulations and Mappings*. NGFI - Next Generation Fronthaul Interface 1914.3. Institute of Electrical and Electronics Engineers (IEEE), June 2018.

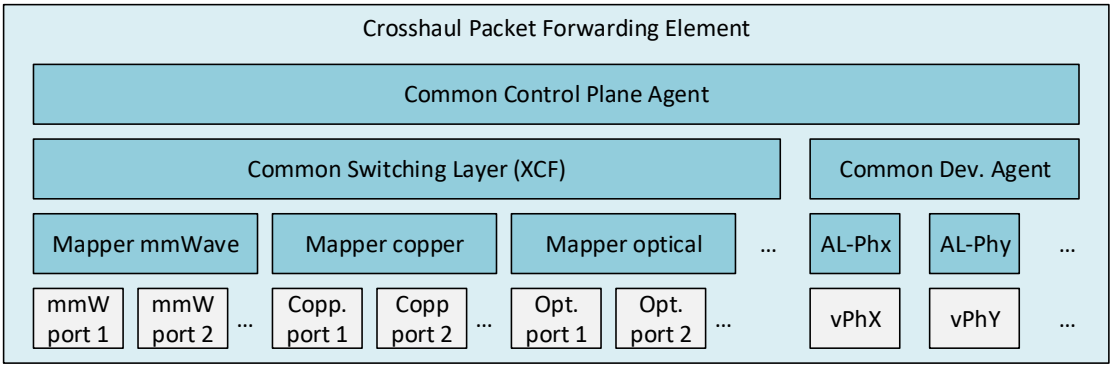


Figure 7.2: Crosshaul Packet Forwarding Element.

7.2.1 Crosshaul multi-layer switch

The Crosshaul Forwarding Element (XFE) is a multi-layer switch for enabling an integrated, flexible, and software-defined reconfigurable data plane for a crosshaul transport network. An XFE may play the role of M1, M2, M3, and M4 nodes in the reference crosshaul architecture illustrated in Figure 7.1. In its generic implementation, the XFE encompasses two macro switching units: the Crosshaul Packet Forwarding Element (XPFE), which deals with packet forwarding, and the Crosshaul Circuit Switching Element (XCSE), which deals with circuit switching. The rationale of the proposed design resides in the fact that a packet-based switch alone may not be suitable for 5G services and interfaces with demanding timing constraints, like CPRI [CPR15]. Similarly, a circuit-based switch alone would not be able to bring the cost efficiency benefits enabled by the statistical multiplexing of packet-based technologies. Therefore, the crosshaul transport network will consist of the interconnection of XFEs integrating heterogeneous physical transmission technologies, either wired or wireless, through a common data frame and forwarding behaviour.

In the case of XCSE, the circuit layer may be further split in two sub-layers having different granularity. In optical networks, the sub-layers correspond to wavelengths and time slots in a wavelength, as in current reconfigurable add drop multiplexers (ROADMs) and according to Time-division Multiplexing (TDM) techniques described in Chapter 7.1.4, respectively. It is not necessary that all the layers always coexist but one or two of them could be skipped depending on the type of deployed network. For example, a mesh network of packet switches connected by dark fibres, i.e., where only the packet layer is exploited; 5G RRHs, based on new radio protocol split and packetised fronthaul interface, connected to a DWDM network, i.e., where both wavelength and packet switches are present; the same network where also CPRI tributaries are carried and multiplexed over time-slots in a wavelength, so that a TDM switch needs to be added. Figure 7.2 depicts the functional architecture for the XPFE, which includes several internal components such as the common device agent, the common switching layer, and the common control-plane agent, which is in charge of the communication with the SDN controller. The device agent is common to all peripherals and exposes to the SDN controller all the device-related information and available operations on the device. For example, the device agent might support several power states, resource slicing, and statistics collection, like CPU usage, RAM occupancy, battery status, Global Positioning System (GPS) position, etc. In order to provide a harmonised view of the device capabilities, an adaptation layer (noted as AL-PhX, AL-PhY, etc.) is introduced between the common device agent and the peripherals (noted as vPhX, vPhY, etc.) as shown in Figure 7.2. Such layer adapts the peripheral-specific parameters to the common peripheral model used by the common device agent.

A key part of the XFE envisioned solution is a common switching layer for enabling a unified and harmonised traffic management. In particular, the switching engine is technology-agnostic and relies on (*i*) an abstract resource model (e.g., bandwidth, latency, BER, jitter, latency, etc.)

of the underlying interfaces (e.g., mmWave, optical, etc.), and on (*ii*) traffic requirements (e.g., fronthaul, backhaul, jitter tolerance, packet loss, etc.). As a result, the common switching layer enables forwarding in the network between heterogeneous protocols (e.g., fronthaul, backhaul), interfaces and physical technologies by leveraging a XCF format. Mappers for each physical interface reside under the common switching layer and are in charge of enforcing the control-plane policies by mapping the commands to protocol and technology-specific interfaces/peripherals. XCF can be mapped on any physical interface as long as the XCF traffic requirements are satisfied. For example, Next Generation Fronthaul Interface (NGFI) digital samples could be carried by XCF and transmitted over a copper interface only if a low-bandwidth-demanding functional split is adopted. If a more demanding functional split is adopted, a different physical interface (e.g., optical, mmWave) is required. This means that multiple physical interfaces can coexist in the unit including different technologies (i.e., fibre optic, mmWave, μWave, copper, etc.). Therefore, the XCF is the transport frame format used within the crosshaul transport network, and it is worth highlighting that it does not impose any constraint on the payload protocol carried within. For the sake of example, let's consider an XPFE with mmWave and fibre optic interfaces under the control of the same common switching layer. The forwarding of XCF frames over multi-technology links is as follows: the XPFE receives a frame over a mmWave link, which employs IEEE 802.11ad [IEE12][24] as MAC layer. Next, the XPFE maps the mmWave frame to XCF and passes it to the common switching layer which, based on the information contained in the XCF, decides how and where to forward the packet. Finally, the XCF is mapped onto a fibre optic frame (e.g., IEEE 802.3av [IEE17a][25]) and sent on the optical link.

7.2.2 Crosshaul common frame

The packet-switching part of the crosshaul data plane needs to support heterogeneous links. To that end, the Crosshaul Common Frame (XCF) should be able to carry both fronthaul and backhaul traffic over such miscellaneous set of links. This section first enumerates the main requirements, both qualitative and quantitative, for the XCF packet technology. Then, it presents the XCF design.

XCF qualitative requirements

The main functional and qualitative requirements for the XCF are:

- Support of multiple functional splits. This includes also the support of backhaul traffic and CPRI-like fronthaul.
- Multi-tenancy. Traffic isolation (e.g., guaranteed QoS), traffic separation (e.g., tenant privacy), differentiation of forwarding behaviour, multiplexing gain, and tenant identification (e.g., identification of tenant's traffic) are features required to enable multi-tenancy;
- Coexistence and compatibility. This includes Ethernet technology, security support, and synchronisation, e.g., IEEE 1588 [IEE08][26], IEEE 802.1AS [IEE10][27];
- Transport efficiency, including short overhead, multi-path support, flow differentiation, and Class of Service (CoS) differentiation;
- Management, including in-band control traffic (e.g., Operations, Administration and Maintenance (OAM));

[24] IEEE. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 3: Enhancements for Very High Throughput in the 60 GHz Band*. Wireless LAN Working Group 802.11ad. Institute of Electrical and Electronics Engineers (IEEE), November 2012.

[25] IEEE. *10Gb/s Ethernet Passive Optical Network*. Standards for Local and metropolitan area networks 802.11av. Institute of Electrical and Electronics Engineers (IEEE), September 2017.

[26] IEEE. *Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. PNCS - Precise Networked Clock Synchronization Working Group 1588. Institute of Electrical and Electronics Engineers (IEEE), 2008.

[27] IEEE. *Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*. Standards for Local and metropolitan area networks 802.1AS. Institute of Electrical and Electronics Engineers (IEEE), November 2010.

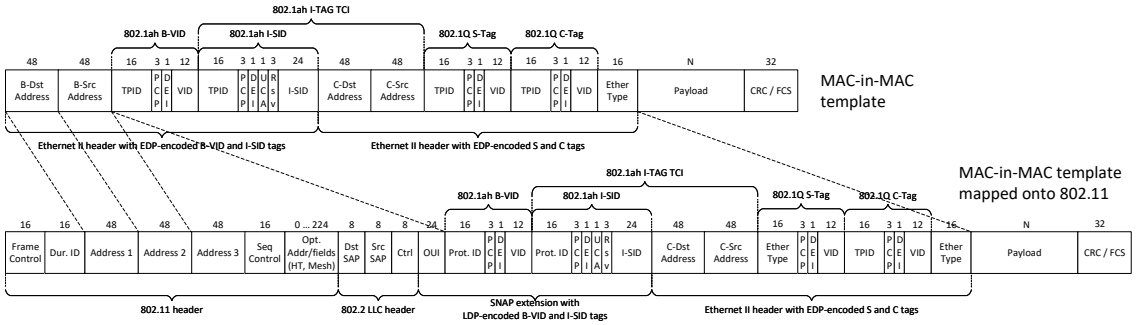


Figure 7.3: Crosshaul Common Frame and its mapping on 802.11 frames.

- Energy efficiency, including energy usage proportional to handled traffic (e.g., sleep mode, reduced rate);
- Support of multiple data link technologies, including Ethernet, IEEE 802.3, IEEE 802.11 (e.g., mmWave), etc.;
- No vendor lock-in.

XCF quantitative requirements

In addition to the qualitative requirements, there are performance and quantitative requirements for the XCF:

- Latency: the maximum end-to-end latency for IQ data between Radio Equipment (REC) and RE Control (REC) must be 100 μs, including the propagation delay of the links between the devices, internal delays of the devices such as bridges. For Control and Managmenet (C&M) there is no latency requirement.
- Frame Loss Ratio (FLR): can be caused by bit error, network congestion, failures, etc. Late delivery can also imply frame loss for CPRI data. It must be less than 10^{-7}. For C&M, the FLR must be less than 10^{-6}.
- Synchronisation. Depending on the type of radio access technology the requirements are different. The maximum absolute time error should be less than 10 ns for intra-band contiguous carrier aggregation radio access technologies. The maximum absolute time error must be less than 45 ns for Multiple-Input and Multiple-Output (MIMO) and transmit diversity radio access technologies. The maximum absolute time error must be less than 110 ns for intra-band non-contiguous and inter-band carrier aggregation radio access technologies. The maximum absolute time error must be less than 110 ns for time division duplex radio access technologies.

The XCF is based on Ethernet frames, therefore relevant standards for packetised fronthaul are summarised before the XCF is presented.

XCF design

Several IEEE working groups have been extending Ethernet to handle time sensitive and fronthaul traffic. Some of the working groups have just started and the proposals are still subject to change. Time Sensitive Networking (TSN) [IEE] is a collection of features in IEEE 802.1, which targets providing extremely low packet loss rates, finite, low and stable end-to-end latency and synchronisation among bridges and end-stations: time synchronisation to an accuracy of better than 1 μs is achieved by IEEE 802.1AS [IEE10]. In such a system, all bridges and end stations have to be time aware. Besides Ethernet (i.e., IEEE 802.3), 802.1AS can also be used, e.g., for Wi-Fi or Ethernet PON. QoS with low latency and high probability of packet delivery is defined in IEEE 802.1CB [IEE17b][28]. It provides redundancy by duplicating packets and sending them on different

[28] IEEE. *Frame Replication and Elimination for Reliability*. Standards for Local and metropolitan area networks 802.1CB. Institute of Electrical and Electronics Engineers (IEEE), September 2017.

paths to a destination. IEEE 802.1Qcc [IEE18d][29] improves the stream reservation protocol, by which end stations can reserve bandwidth for traffic stream. IEEE 802.1Qbv [IEE15b][30] extends the credit-based shaper by operations to open or close the gate for a traffic class. High priority traffic can be sent as soon as its gate opens, i.e., it can be sent with less jitter. IEEE 802.1Qci [IEE17c][31] provides protection against systems that could disrupt QoS or time synchronisation by filter mechanisms based on packet rate and packet size at the boundary of a domain. This prevents misbehaving nodes to flood a network with too many or too large packets which in turn would break guarantees on latency and jitter. IEEE 802.1Qbu [IEE15c][32] allows express frames to preempt other frames. This reduces jitter for the express traffic. IEEE 802.1CM [IEE18e] defines a profile to exploit these features to satisfy the latency requirements for fronthaul traffic, especially those of CPRI data. IEEE 1914.3 [IEE18b] defines how CPRI data is mapped to Ethernet frames. Ethernet frames with CPRI data are indicated by a dedicated EtherType. The header of CPRI over Ethernet is part of the Ethernet payload. Two different mappings for CPRI data are defined: the structure agnostic mapping does not interpret the CPRI data in any way, it just puts the CPRI data to the Ethernet frame. The structure aware mapping differentiates between actual antenna data and, e.g., control data and sends them in different Ethernet frames, which potentially may have different priority markings. Frame preemption (i.e., 802.1Qbu) is considered beneficial, where possible for the specific data link, also frame filtering (i.e., 802.1Qci) to guard the network against malfunctioning nodes is considered beneficial.

Circuit and packet switching may be simultaneously deployed in a crosshaul network. Even for packet switching alone, heterogeneous technologies may be deployed, such as Ethernet (IEEE 802.3), mmWave radio, Wi-Fi (IEEE 802.11), etc. The XCF is used to transport data across such heterogeneous links. The XCF is required to support multiple functional splits of the radio protocol stacks, i.e., traffic streams with diverse characteristics have to be transported in such frames. The XCF has to support prioritisation as well as to prevent that these diverse traffic streams influence each other. The traffic streams may belong to different operators. Therefore, the XCF has to support multi-tenancy, keeping the streams of different tenants separate and allowing different forwarding behaviour per tenant. The XCF has to be efficient, it should have low protocol overhead, allow the use of multiple paths towards one destination, and allow for multiplexing gains. Finally, the XCF needs to be compatible with legacy technology and it is required to carry synchronisation information. Considering all the above requirements, a frame based on Ethernet has been considered as the best option for the XCF, more specifically on Provider Backbone Bridges (PBB) frames, also known as MAC-in-MAC [IEE09c]. Basing the XCF on Ethernet allows using all the forthcoming extensions as described before. PBB frames provide multi-tenancy support: the infrastructure provider can deploy multiple virtual networks and a tenant can provide additional separation within its network. The priority code points allow encoding different priorities, relevant to separate the types of traffic classes from each other. Eventually, a flow id can be used to direct different flows with same destination to different paths, while keeping each individual flow on the same path. The XCF can be used as well on Wi-Fi based transport links, using the proposed standard amendment IEEE 802.11ak [IEE17e][33].

[29] IEEE. *Stream Reservation Protocol (SRP) Enhancements and Performance Improvements*. Standards for Local and metropolitan area networks 802.1Qcc. Institute of Electrical and Electronics Engineers (IEEE), June 2018.

[30] IEEE. *Enhancements for Scheduled Traffic*. Standards for Local and metropolitan area networks 802.1Qbv. Institute of Electrical and Electronics Engineers (IEEE), October 2015.

[31] IEEE. *Per-Stream Filtering and Policing*. Standards for Local and metropolitan area networks 802.1Qci. Institute of Electrical and Electronics Engineers (IEEE), September 2017.

[32] IEEE. *Frame Preemption*. Standards for Local and metropolitan area networks 802.1Qbu. Institute of Electrical and Electronics Engineers (IEEE), October 2015.

[33] IEEE. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 4: Enhancements for Transit Links Within Bridged Networks*. Wireless LAN Working Group 802.11ak. Institute of Electrical and Electronics Engineers (IEEE), September 2017.

Table 7.1: Traffic flow composition per network slice, activity factor, and service area dimension.

SLICE	TRAFFIC FLOW	C (GBPS/KM^2)		α	D		
		Downlink	Uplink		Urban	Industrial	Rural
eMBB	Urban macro	100	50	0.200	1.000	1.000	0.000
	Rural macro	1	0.5	0.200	0.000	0.000	1.000
	Indoor hotspot	15000	2000	1.000	0.030	0.000	0.000
	Dense urban	750	125	0.100	0.040	0.000	0.000
	Broadcast services	20	0	1.000	1.000	0.000	0.000
	High-speed vehicle	700	50	0.500	0.020	0.000	0.016
URLLC	Motion control	1000	1000	1.000	0.000	0.010	0.000
	Discrete automation	1000	1000	1.000	0.000	0.010	0.000
	Remote control	100	100	1.000	0.000	0.090	0.000
	Process monitoring	10	10	1.000	0.000	0.090	0.000
	Elec. distr.: Med. V	10	10	1.000	0.020	0.010	0.001
	Elec. distr.: High V	100	100	1.000	0.010	0.010	0.001
	Intel. transport system	10	10	1.000	0.032	0.032	0.032
MIoT	Massive MTC	0	200	0.100	1.000	0.090	0.010

α : Activity Factor; C : Area Traffic Capacity per Area Unit; D : Service Area Dimension Factor;

7.3 Simulating a crosshaul network for understanding QoS applicability

The goal of this section is to better understand how packet-based multiplexing can be applied in the crosshaul network in order to satisfy the 5G network slices traffic requirements, namely eMBB, URLLC, and MIoT. Specifically, the analysis focuses on how different queueing policies (e.g., FIFO, Strict Priority, etc.) and configurations (e.g., traffic marking) help at fulfilling the target service requirements.

7.3.1 Deriving the network slices requirements for the reference architecture

The reference architecture illustrated in Figure 7.1 identifies the number of hops and multiplexing points in the network. In order to perform the desired simulation, it is necessary to identify and define first the proper traffic flow mixture and network slices the crosshaul network needs to transport. To that end, a series of flows with the corresponding traffic requirements for the eMBB and URLLC slices is defined by 3GPP in [3GP18j]. A set of traffic requirements for the MIoT slice is defined instead by NGMN in [NGM15a][34].

Table 7.1 presents an integrated version of the original separated tables[3GP18j], [NGM15a]. It is worth highlighting that this table does not include those flows that are expected to have a limited or ad-hoc deployment, such as broadband access in a crowded event (e.g., limited to stadium/venues), high-speed train (e.g., only along the railway), airplanes connectivity (e.g., sporadic terrestrial base stations), and those flows whose requirements are not fully defined yet, such as tactile interaction and remote control. Each traffic flow is provided with an Area Traffic Capacity, both in Uplink and Downlink, which identifies the expected traffic density expressed in Gbps/km^2. Next, an Activity Factor indicates the percentage of time the devices generating that type of traffic are expected to be alive. Finally, a Service Area Dimension specifies as scaling factor the percentage of space a given flow is considered to be present. Therefore, the total amount of bandwidth (Gbps) required per area unit (km^2) is given by Equation 7.1.

Definition 7.1: Total traffic per area unit.

$$\Sigma = \alpha \times C \times D \tag{7.1}$$

[34] NGMN. *5G White Paper*. White Paper v1.0. Next Generation Mobile Networks Alliance, February 2015.

Σ : total traffic per area unit (Gbps/km^2); α : activity factor;
C : area traffic capacity per area unit (Gbps/km^2); D : service area dimension factor;

Given the considerably large distance traversed by the transport network (up to 100 Km), a wide variety of scenarios is expected to be present in terms of traffic characterisation. To reflect this aspect, three scenarios are defined: urban, industrial, and rural. An urban scenario is exemplified by a city environment where dense connectivity, intelligent transportation system, high-speed vehicle, and indoor hot-spots are present. In turn, an industrial scenario is characterised by the presence of URLLC traffic related to discrete automation. Finally, the rural scenario considers an open environment with a trafficked 4-lane road. For the sake of clarity, a given scenario considers a specific traffic flow only when the Service Area Dimension is non-zero in Table 7.1. The next step is to calculate the number of AAUs required to cover an area unit (1 km^2). Additional considerations are required in this case given to the different physical deployments. For instance, a service dimension area for the indoor hotspot is 0.040. This result is obtained by considering a tall building with 15 floors with a base of 0.2×0.2 km^2. The total bandwidth to provide would be 600 Gbps spread over the 15 floors, thus 40 Gbps on each floor. As a result, 4 AAUs per floor are required, considering a peak data rate of 10 Gbps for the AAU. A total of 72 AAUs are hence required to cover 1 km^2 in an urban scenario. A similar consideration is made for the industrial scenario where the service area dimension is considered of either 0.01 km^2 or 0.09 km^2 depending on the type of traffic. A total of 12 AAUs is hence required to satisfy the traffic demand of 1 km^2 in an industrial scenario. Finally, for the rural scenario, a 4-lane road 16 meters wide is considered resulting in a surface of 0.016 km^2. Subsequently, 1 AAU provides enough capacity to cover 1 km^2 in a rural scenario.

Table 7.2: Traffic load per AAU per scenario.

Scenario	Σ (Gbps/km^2)		#AAU/km^2	Traffic load/AAU (Gbps)	
	Downlink	Uplink		Downlink	Uplink
Urban	501.52	92.52	72	6.97	1.25
Industrial	51.32	43.12	12	4.27	3.59
Rural	6.23	1.13	1	6.23	1.13

Σ : Total traffic per area unit;

Table 7.3: Traffic flow load per AAU for the different scenarios and slices.

Slice	Traffic Flow	Urban (Mbps)		Industrial (Mbps)		Rural (Mbps)	
		Downlink	Uplink	Downlink	Uplink	Downlink	Uplink
eMBB	Urban macro	277.78	138.89	1666.67	833.33	0.00	0.00
	Rural macro	0.00	0.00	0.00	0.00	200.00	100.00
	Indoor hotspot	6250.00	833.33	0.00	0.00	0.00	0.00
	Dense urban	41.67	6.94	0.00	0.00	0.00	0.00
	Broadcast services	277.78	0.00	0.00	0.00	0.00	0.00
	High-speed vehicle	97.22	6.94	0.00	0.00	5600.00	400.00
URLLC	Motion control	0.00	0.00	833.33	833.33	0.00	0.00
	Discrete automation	0.00	0.00	833.33	833.33	0.00	0.00
	Remote control	0.00	0.00	750.00	750.00	0.00	0.00
	Process monitoring	0.00	0.00	175.00	175.00	0.00	0.00
	Elec. distr.: Med. V	2.78	2.78	8.33	8.33	10.00	10.00
	Elec. distr.: High V	13.89	13.89	83.33	83.33	100.00	100.00
	Intel. transport system	4.44	4.44	26.67	26.67	320.00	320.00
MIoT	Massive MTC	0.00	277.78	0.00	150.00	0.00	200.00

Table 7.2 reports those results in addition to the average load experienced by each AAU in each scenario. In turn, Table 7.3 reports the bandwidth breakdown for a single AAU for each traffic flow

Table 7.4: Packet size for each traffic flow.

SLICE	TRAFFIC FLOW	DOWNLINK	UPLINK
eMBB	Urban macro	See Figure 7.4 (Mobile)	See Figure 7.4 (Mobile)
	Rural macro	See Figure 7.4 (Mobile)	See Figure 7.4 (Mobile)
	Indoor hotspot	See Figure 7.4 (Fixed)	See Figure 7.4 (Fixed)
	Dense urban	See Figure 7.4 (Mobile)	See Figure 7.4 (Mobile)
	Broadcast services	1374 bytes [Fow+14]	No uplink traffic
	High-speed vehicle	See Figure 7.4 (Mobile)	See Figure 7.4 (Mobile)
URLLC	Motion control	255 bytes [3GP18j]	255 bytes [3GP18j]
	Discrete automation	1358 bytes [3GP18j]	1358 bytes [3GP18j]
	Remote control	160 bytes [3GP18j]	160 bytes [3GP18j]
	Process monitoring	640 bytes [3GP18j]	640 bytes [3GP18j]
	Elec. distr.: Med. V	128, 256, 512, 1024 bytes [WD14]	128, 256, 512, 1024 bytes [WD14]
	Elec. distr.: High V	128, 256, 512, 1024 bytes [WD14]	128, 256, 512, 1024 bytes [WD14]
	Intel. transport system	320 bytes [3GP18j]	320 bytes [3GP18j]
MIoT	Massive MTC	No downlink traffic	94, 144, 234, 327, 699 bytes [Siv+17]

and scenario. One additional consideration is required for understating how the AAUs for each scenario are deployed in the network. Given the expected physical deployment of the AAUs, as well as the expected traffic mix, the access rings are considered to be separated for each scenario. That is, the traffic mix present on any AAU on the same access ring is the same (e.g., urban, rural, industrial). Furthermore, the composition of access rings connecting to a M2 node is the following: 2 urban, 1 industrial, 1 rural. Summarising, at M1 level the traffic multiplexed is separated per scenario (e.g., there is no urban and industrial traffic multiplexing), while at M2 and M3 level traffic belonging to different scenarios is instead multiplexed. The last traffic characterisation relevant for the QoS analysis is the packet size distribution of each traffic flow. Indeed, the length of each packet influences the transmission delay and the jitter as result of the interaction between packets in the queue.

Table 7.4 reports the packet size distribution for each traffic flow, which is derived from both experimental observation of a real operator network[35] and reference values. Specifically, the packet size distributions obtained from real traffic captures in a mobile and a fixed network are shown in Figure 7.4. The data measured in a mobile scenario is used to characterise the urban macro, rural macro, dense urban, and high-speed vehicle traffic flows. It is worth highlighting that traffic generated by high-speed vehicles in this case belong to the eMBB slice, which identifies the multimedia traffic (e.g., generated by people in the car or car infotainment). In the case of downlink, the packet size distribution presents a minor peak at 64 bytes while the most prominent peak is present at 1450 bytes. The uplink shows the main peak at 64 bytes. The data measured in a fixed network is used instead for characterising the indoor hotspot traffic flow which is considered to have a similar traffic pattern to fixed access due to the stationary nature of the users. The downlink presents a minor peak at 64 bytes with the highest peak at 1480 bytes. On the other hand, the uplink presents a minor peak at 1450 bytes whilst the main peak is at 64 bytes. Regarding the broadcast services traffic flow, the reference value is available in [Fow+14][36]. Moreover, the packet size for the motion control, discrete automation, remote control, process monitoring, and intelligent transport system traffic flows are specified by 3GPP [3GP18j]. Compared to the high-speed vehicle traffic flow, intelligent transport systems belongs to the URLLC slice and encompasses the Vehicle-to-Infrastructure (V2I) traffic. Reference values for the electricity distribution traffic flows are

[35] Aggregated and anonymised data for the packet size distribution has been kindly provided by Telefonica. The data report the national fixed and mobile traffic generated in May 2017.

[36] S. Fowler et al. 'Evaluation and prospects from a measurement campaign on real multimedia traffic in LTE vs. UMTS'. in: *2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace Electronic Systems (VITAE)*. May 2014, pages 1–5. DOI: 10.1109/VITAE.2014.6934475.

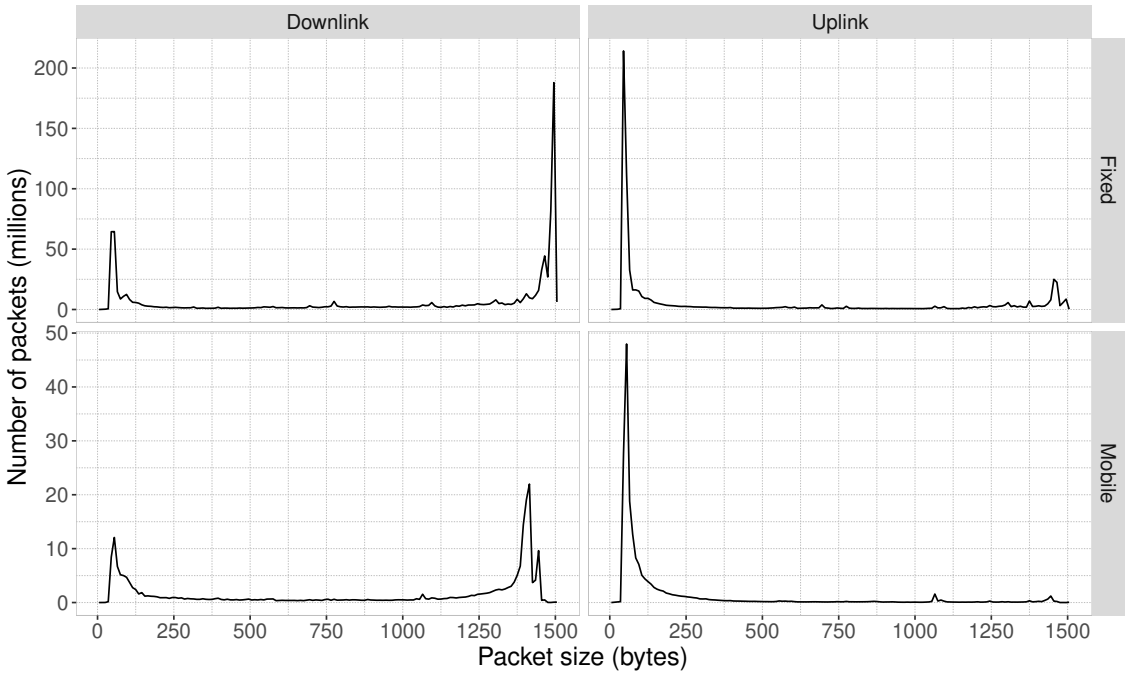


Figure 7.4: Packet size distribution for fixed and mobile scenarios.

available in [WD14][37], while for the massive Machine Type Communication (MTC) are available in [Siv+17][38].

7.3.2 Simulation scenario, framework, and results

Based on the reference architecture (see Chapter 7.1.2), the various Ethernet-based transport technologies (see Chapter 7.1.3), and the expected mixture of network slices traffic (see Chapter 7.3.1), the simulation scenario illustrated in Figure 7.5 is derived. It is worth highlighting that the the link speeds reported in Figure 7.5 are the result of traffic requirement calculations and are not provided in the reference document [ITU18b]. To analyse the impact of different queueing disciplines in the crosshaul network, a simulation framework based on SimPy, namely SimPype, has been developed and published as open-source[39]. SimPype[40] relies on the concepts of resource and pipe, and decouples the resource from its queue (pipe) in such a way that multiple queueing techniques can be used with the same resource. For this reason, SimPype is well-suited to simulate scenarios where the queueing disciplines and the resources occupation are key parts of the system (e.g., packet-based network). SimPype also allows to create both custom resource and pipe models that can be reused in multiple simulations. In particular, in our QoS analysis we considered the following queueing policies: First-In, First-Out (FIFO), Strict Priority, and Strict Priority with Preemption. A FIFO queue is the simplest type of queue where all the packets have the same priority and the decision of which packet needs to be transmitted next is only based on the time of arrival of the packet. For this reason, a FIFO queue is also known as First-Come, First-Served (FCFS). In turn, a Strict Priority policy assigns a priority to each packet to be transmitted, meaning that the selection of the packet is based on the priority rather than on the time of arrival. As a result, packets with higher priority are always transmitted first. Lastly, the Strict Priority with Preemption

[37] T. J. Wong and N. Das. 'Modelling and analysis of IEC 61850 for end-to-end delay characteristics with various packet sizes in modern power substation systems'. In: *5th Brunei International Conference on Engineering and Technology (BICET 2014)*. November 2014, pages 1–6. DOI: 10.1049/cp.2014.1073.

[38] A. Sivanathan et al. 'Characterizing and classifying IoT traffic in smart cities and campuses'. In: *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. May 2017, pages 559–564. DOI: 10.1109/INFCOMW.2017.8116438.

[39] SimPype: http://simpype.readthedocs.io/en/latest/

[40] An in-depth description of SimPype concepts and some examples can be found in Appendix A.

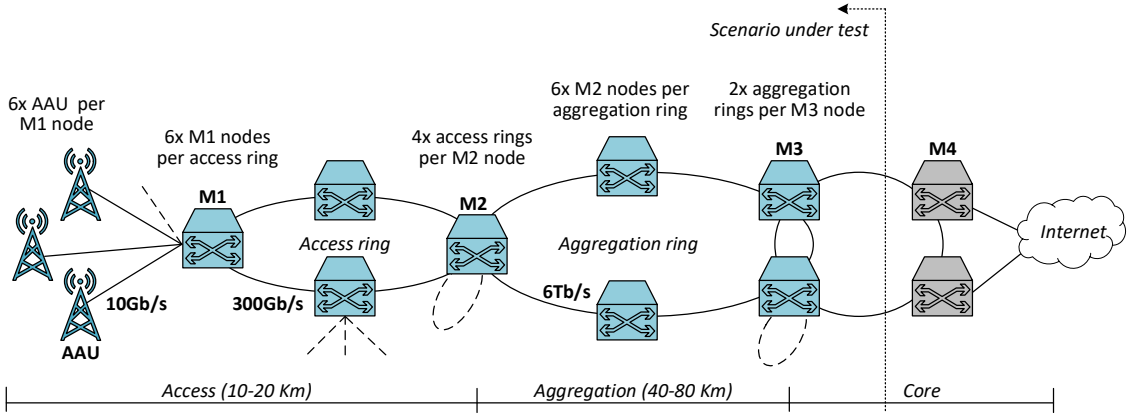


Figure 7.5: Scenario under simulation for understanding packet multiplexing in a crosshaul network transporting eMBB, URLLC, and MIoT network slices.

policy works in an analogous way to Strict Priority with the exception that a packet with highest priority can preempt an ongoing transmission of a packet with lower priority. That means that the transmission of a low priority packet is suspended in favour of the transmission of a higher priority packet. The transmission of the low priority packet is then resumed only when the higher priority packet is successfully transmitted. Packet preemption in Ethernet networks is defined in [IEE15c].

The simulation implements the network scenarios and the traffic flow reported in Chapter 7.3.1 while considering a failure-free environment where the protection rings are not activated. We preformed 100 simulations of a 10 ms time frame across all the transport network including the generation and transmission of $\sim$6 million packets. For the access and aggregation segments we considered a distance of 15 km and 60 km, respectively, with a transmission delay of 5 μs/km [Com17][41]. Regarding the priority and preemption policies, packets are prioritised according to their slice with the URLLC having the highest priority and MIoT the lowest. Moreover, preemption is allowed only for URLLC packets against eMBB and MIoT packets whilst eMBB packets do not preempt MIoT ones. Finally, packet generation follows an exponential arrival time. Figure 7.6 reports the simulation results for the considered scenarios, traffic flows, and queueing disciplines both in uplink and downlink. The depicted box plots report the 5$^{\text{th}}$, 25$^{\text{th}}$, 50$^{\text{th}}$, 75$^{\text{th}}$, and 95$^{\text{th}}$ percentile of the end-to-end delay, that is from AAU to M3 for uplink and from M3 to AAU in downlink. As it can be noticed, the largest delay component is the transmission delay (75 km $\times$ 5 μs/km = 375 μs). The queueing contributes with an additional delay between 0.05 μs and 4.91 μs depending on the traffic flow. For sake of simplicity, we did not consider any additional delay introduced by any processing time at switch level. The traffic flow with the minimum delay is intelligent transport systems in urban scenario with the strict priority with preemption queueing policy. In turn, the traffic flow with maximum value is MIoT in industrial scenario with the strict priority with preemption queueing policy. Comparing the results with the traffic requirements defined in [3GP18i], all the traffic flows fulfil the requirements in terms of delay and jitter except the motion control in industrial scenario. Specifically, 3GPP defines a maximum jitter of 1 μs for the remote control of actuators in industrial robots which is not satisfied in the scenario under test.

Table 7.5 reports the jitter values in μs for the motion control traffic flow for different configurations. The first configuration, namely M3 URLLC Unified, considers URLLC traffic traversing all the network from the AAU to M3 nodes (and vice versa) and all URLLC packets have the same priority regardless the traffic flow they belong to. This configuration is labelled as M3 URLLC Unified in Table 7.5. As it can be noticed, the minimum jitter in downlink (1.949 μs) and uplink (2.575 μs) is achieved with the strict priority with preemption queueing discipline. However, such value exceeds the maximum admissible value of 1 μs. Therefore, we performed additional simulations as to find a configuration capable of fulfilling the traffic requirements. The following

[41] CommScope. *Latency in optical fiber systems*. White Paper. 2017.

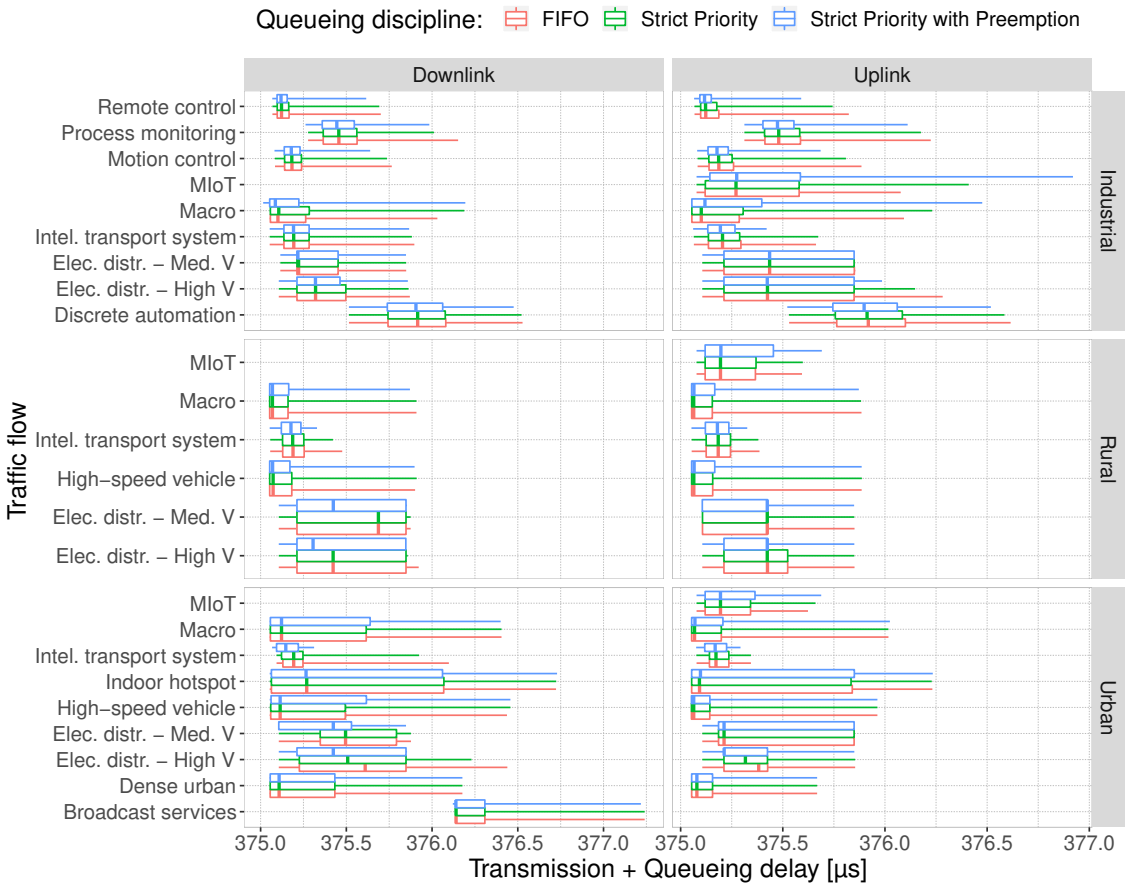


Figure 7.6: End-to-end transmission and queueing delay for different traffic flows being terminated at M3 nodes.

Table 7.5: Jitter for motion control traffic flow terminated at M1, M2, and M3 nodes under different URLLC multiplexing configurations.

Configuration	Policy	Jitter (μs)	
		Downlink	Uplink
M3 URLLC Unified	FIFO	1.957	3.152
	Strict Priority	1.957	2.575
	Strict Priority with Preemption	1.949	2.575
M2 URLLC Unified	FIFO	2.250	4.125
	Strict Priority	2.250	2.704
	Strict Priority with Preemption	2.250	2.661
M2 URLLC Separated	FIFO	2.250	4.125
	Strict Priority	1.582	1.447
	Strict Priority with Preemption	0.510	1.117
M1 URLLC Unified	FIFO	2.364	2.375
	Strict Priority	2.120	1.951
	Strict Priority with Preemption	2.120	1.951
M1 URLLC Separated	FIFO	2.670	2.556
	Strict Priority	1.370	1.304
	Strict Priority with Preemption	0.549	0.446

scenario under test terminates all the URLLC traffic at M2 level meaning that URLLC traffic does not traverse M3 nodes neither in downlink or uplink. In the first configuration of this new scenario (labelled as M2 URLLC Unified in Table 7.5), traffic flows belonging to the URLLC slice have

the same priority. As it can be noticed, even terminating the motion control at M2 level does not allow to comply with the traffic requirements. Indeed, the minimum jitter in downlink (2.250 μs) and uplink (2.661 μs) is achieved with the strict priority with preemption queueing discipline. It is worth highlighting that downlink jitter in this configuration is higher compared to the one obtained by terminating the traffic at M3 level. This is because packets in downlink are enqueued and transmitted at higher speed at M3 level and arrive already sorted at M2 nodes which do not need to perform additional prioritisation at lower speed.

As a next step, the motion control traffic flow is treated differently in the URLLC slice. That is, a higher priority is given to the motion control traffic flow with regards to the other URLLC traffic flows (labelled as M2 URLLC Separated in Table 7.5). By doing so, such flow gains the highest priority in the whole network. As a result, the 1 μs jitter is satisfied only in downlink(0.510 μs) when the strict priority with preemption queueing discipline is used. However, the 1 μs jitter requirement is not fulfilled in uplink (1.117 μs). Consequently, an additional simulation scenario is considered where all the URLLC traffic at M1 level meaning that URLLC traffic does not traverse M2 and M3 nodes neither in downlink or uplink. In the first configuration of this new scenario (labelled as M1 URLLC Unified in Table 7.5), traffic flows belonging to the same slice (i.e., URLLC) have the same priority. As it can be noticed, even terminating the motion control at M1 level does not allow to comply with the traffic requirements. Indeed, the minimum jitter in downlink (2.120 μs) and uplink (1.951 μs) is achieved with the strict priority with preemption queueing discipline. Similarly to the M2 case, the downlink jitter in this configuration is higher compared to the one obtained by terminating the traffic at M3 level. The last configuration considers assigning a higher priority to the motion control compared to the other URLLC flows (labelled as M1 URLLC Separated in Table 7.5). Results show that the target jitter is fulfilled only when the strict priority with preemption queueing discipline is used and results in a jitter of 0.549 μs in downlink and 0.446 μs in uplink. Therefore, it is advisable in case of a 5G transport network to terminate the URLLC motion control traffic flow at most at M1 nodes to fulfil the corresponding traffic requirements.

8. Monitoring a SDN-based 5G transport network

Software Defined Networking (SDN)-based architectures potentially enable an optimal management of the network. This is true as long as the applications operating the SDN network, which are running on a logically-centralised Network Controller (NC), are timely provided with a rich set of statistics collected from the underlying infrastructure. The predominant SDN standard for the Southbound Interface (SBI), which enables the communication between the infrastructure and the controller, is OpenFlow. However, OpenFlow was originally conceived for campus and data centre networks, therefore it was designed with a different set of requirements in mind compared to mobile networks. Although the initial limited set of functionalities has been gradually extended to cover new protocols (e.g., Multiprotocol Label Switching (MPLS), Provider Backbone Bridges (PBB)) and more sophisticated forwarding behaviours typical of mobile networks (e.g., packet encapsulation), monitoring support in OpenFlow is still limited compared to what is required [IEE09a][1], [ITU15c][2]. Considering the applicability to mobile networks, one of the areas where OpenFlow lags behind is monitoring and fault-detection [Ber+14].

OpenFlow, as currently defined, does not really support rapid and scalable monitoring of resources. As a matter of fact, OpenFlow only permits the network controller to poll the switches for gathering simple statistics (e.g., number of packets, transmitted/received bytes, etc.) and requires any additional measurement to be directly implemented on top of the network controller. This is because of the contrasting design principles adopted by Operations, Administration and Maintenance (OAM) and SDN:

- OAM defines *stateful* mechanisms that must be executed on the switch.
- OpenFlow defines a *stateless* forwarding model for the switch and delegates stateful logic to the controller.

As a consequence, realising an SDN-based monitoring presents significant challenges both in terms of scalability and accuracy in mobile networks. Indeed, the network controller needs to directly perform the necessary measurements on each of the numerous (up to tens of thousands) and distant (up to hundreds of kms) network nodes and links with the required precision and granularity (up to microseconds) in order to provide the necessary reliability (up to 99.9999%) to the very distinct services in 5G [3GP18i]. To overcome those issues, the current SDN-based monitoring needs to be augmented with an automated communication process, namely *telemetry*, by which measurements

[1] IEEE. *Connectivity Fault Management*. Standards for Local and metropolitan area networks 802.1ag. Institute of Electrical and Electronics Engineers (IEEE), November 2009.

[2] ITU-T. *Operations, administration and maintenance (OAM) functions and mechanisms for Ethernet based networks*. Series G: Transmission Systems and Media, Digital Systems and Networks, Series Y: Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks G.8013/Y.1731. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), August 2015.

and other data are: (*i*) generated and collected locally at network nodes subject to different service requirements, and (*ii*) transmitted to the controller for enabling an optimal network management.

Many works in literature have shown the benefits of applying the SDN paradigm to network management, resulting in a better Quality of Service (QoS) for various applications and services. Particular attention has been given to the routing algorithms capable of selecting the best path for different types of traffic. For instance, Tomovic et al. [TPR14][3] consider two classes of traffic: priority traffic, with strict bandwidth requirements, and best-effort traffic. The proposed frameworks calculates the optimal route as the shortest route having sufficient *bandwidth* available while minimising degradation of best-effort traffic. Egilmez et al. [ECT13][4] propose instead an analytical framework for optimising the forwarding of QoS-enabled streaming of scalable encoded videos. The optimal routes are calculated by solving a constrained shortest path problem where the *jitter* of the available paths is provided as input. Moreover, a variant of the proposed algorithm is proposed for interactive multimedia applications where the total *delay* is considered instead of the jitter. Tomovic et al. [TRP15][5] analyse the suitability of different routing algorithms for performance-guaranteed traffic tunnels in backbone SDN networks subject to two constraints: *bandwidth* and path *delay*. The analysis considered both the computational time on the SDN controller and the bandwidth rejection ratio, which is commonly used as performance indicator for QoS routing algorithms. Similarly, Guck et al. [Guc+18][6] provide a comprehensive evaluation framework and quantitative comparison of centralised QoS routing algorithms in SDN networks subject to *bandwidth*, *delay*, and *jitter* constraints. Finally, Bari et al. [Bar+13][7] propose PolicyCop, a vendor-agnostic QoS policy management framework for OpenFlow-based SDN. The framework provides an interface for specifying QoS-based Service Level Agreement (SLA) for *bandwidth*, *latency*, and *reliability guarantees*, and enforces them using OpenFlow capabilities.

8.1 Current OAM protocols in a glimpse

The most widely-employed transport protocols nowadays are MPLS Transport Profile (MPLS-TP) [Boc+10] and PBB Traffic Engineering (PBB-TE) [IEE09b]. The OAM tool-sets of those protocols are based on ITU-T Y.1731 [ITU15c][8] and IEEE 802.1ag [IEE09a][9] standards, respectively, and they present largely identical characteristics, being the former a superset of the latter. Both protocols define *Connectivity Fault Management (CFM)* mechanisms for path discovery, fault detection, fault notification, fault recovery, fault verification and isolation. In addition, Y.1731 defines OAM functions for *performance monitoring*, such as frame loss, frame delay, and throughput measurements. These functions are typically implemented through the set of protocols

[3] S. Tomovic, N. Prasad and I. Radusinovic. 'SDN control framework for QoS provisioning'. In: *2014 22nd Telecommunications Forum Telfor (TELFOR)*. November 2014, pages 111–114. DOI: 10.1109/TELFOR.2014.7034369.

[4] H. E. Egilmez, S. Civanlar and A. M. Tekalp. 'An Optimization Framework for QoS-Enabled Adaptive Video Streaming Over OpenFlow Networks'. In: *IEEE Transactions on Multimedia* 15.3 (April 2013), pages 710–715. ISSN: 1520-9210. DOI: 10.1109/TMM.2012.2232645.

[5] S. Tomovic, I. Radusinovic and N. Prasad. 'Performance comparison of QoS routing algorithms applicable to large-scale SDN networks'. In: *IEEE EUROCON 2015 - International Conference on Computer as a Tool (EUROCON)*. September 2015, pages 1–6. DOI: 10.1109/EUROCON.2015.7313698.

[6] J. W. Guck et al. 'Unicast QoS Routing Algorithms for SDN: A Comprehensive Survey and Performance Evaluation'. In: *IEEE Communications Surveys Tutorials* 20.1 (April 2018), pages 388–415. ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2749760.

[7] M. F. Bari et al. 'PolicyCop: An Autonomic QoS Policy Enforcement Framework for Software Defined Networks'. In: *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*. November 2013, pages 1–7. DOI: 10.1109/SDN4FNS.2013.6702548.

[8] ITU-T. *Operations, administration and maintenance (OAM) functions and mechanisms for Ethernet based networks*. Series G: Transmission Systems and Media, Digital Systems and Networks, Series Y: Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks G.8013/Y.1731. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), August 2015.

[9] IEEE. *Connectivity Fault Management*. Standards for Local and metropolitan area networks 802.1ag. Institute of Electrical and Electronics Engineers (IEEE), November 2009.

described in the following.

8.1.1 Continuity Check

This protocol comprises the periodical transmission of heartbeat messages, namely *Continuity Check Message (CCM)*, to detect connectivity failures. These messages do not solicit a response and their transmission rate can be configured according to 7 standard values, spanning from 300 messages per second to 6 messages per hour[10]. A sequence number can be optionally used to count CCM losses and detect any eventual link degradation. A burst of Continuity Check messages can be used for measuring one-way *bandwidth*, i.e., on asymmetric links. When the clock of the switches is synchronised, CCM messages can be timestamped and used for measuring the one-way *delay*.

8.1.2 Loopback

This protocol is used for fault verification and isolation. Loopback messages comprise *Loopback Message (LBM)* and *Loopback Reply (LBR)*, which are similar in concept to the request/reply of the *ping* tool. By sending Loopback messages to successive network nodes, an operator can determine the location of a fault as well as a measurement of the two-way frame *delay* and *jitter* in a given network segment. Measuring the two-way delay with Loopback messages does not require the clocks of the switches to be synchronised. A burst of Loopback messages can be used for measuring two-way *bandwidth* on symmetric links.

8.1.3 Link Trace

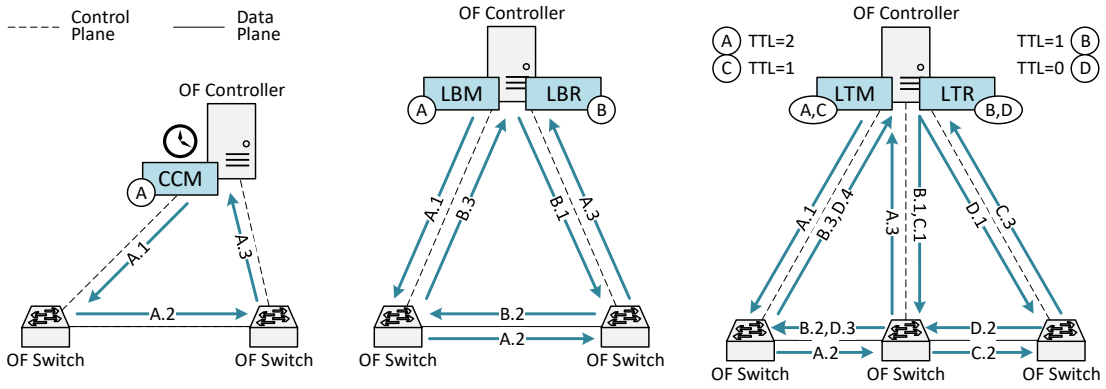
This protocol is used for on-demand path discovery and verification between a pair of network nodes. A *Linktrace Message (LTM)* traverses hop-by-hop every node along the path between a source and a target node, the Time to Live (TTL) of each message is increased until it reaches the destination node (this is similar in concept to the *traceroute* tool). Each hop responds the request messages with a *Linktrace Reply (LTR)* back to the originating node thus allowing the operator to track the path followed by the initial message.

8.2 Analysis of monitoring techniques in current SDN solutions

Monitoring support is very limited in OpenFlow. Current network controllers perform network monitoring by keeping track of the status of the OpenFlow ports by periodically collecting port statistics from the switches [ECT13], [Guc+18], [TPR14], [TRP15]. These statistics provide information regarding the number of packets sent/received or dropped by the port, and whether the port is alive or not. Collecting those statistics involves a message exchange between the network controller and the switch that weights $\sim$600 bytes for a single port. According to the reference architecture shown in Figure 7.1 (see Chapter 7), there are $\sim$200 000 ports for the network segment spanning from the antenna sites to the core ring. This results in a $\sim$1 Gbps of monitoring traffic in case of polling the port statistics every second in that network segments. In the example cited in Chapter 7 for Germany [Nau+], there are 12 of those segments, resulting in a total of $\sim$12 Gbps. Nonetheless, this is only the bandwidth required to collect the port statistics which do not include any information related to the traffic flows configured in the network. To retrieve these statistics, the network controller needs to periodically poll the switches with a message exchange of $\sim$500 bytes for each flow and switch. In case of having various flows configured in the network, the bandwidth required for monitoring can easily grow up to tenths of Gbps.

The periodic polling of statistics presents three main drawbacks in mobile networks: (*i*) a huge amount of bandwidth is required in the control infrastructure, (*ii*) the network controller should be able to process this huge amount of monitoring information in time, and (*iii*) the granularity offered to applications is bound to the polling interval, therefore an application would not be able to react

[10] This analysis focuses on the 3 highest transmission rates, which result in a inter-message interval of 3.3 ms, 10 ms, and 100 ms, respectively.



(a) Continuity Check (b) Loopback (c) Linktrace

Figure 8.1: SDN-based operation of current OAM protocols.

quicker than the configured polling time. Additionally, current OpenFlow statistics do not provide the network controller with sufficient information for knowing the current status of the ports and links (e.g., delay, jitter, available *vs* advertised bandwidth, congestion level, etc.). For instance, a port may be considered alive but may not have link layer connectivity because of misconfiguration (e.g., wrong VLAN). Indeed, an OpenFlow port is considered alive if the carrier at physical level is detected while no additional information is available on the status of the link itself. To overcome such limitations, a set of active measurements is required to enable fine grain knowledge of the network status.

8.2.1 Active monitoring with OpenFlow switches

Active monitoring rely on the capability to inject test packets into the network, following them, measuring the relevant metrics, and store the results for future network optimisation or data analytics. A key feature of being *active* is hence the capability of controlling the nature of the traffic generation, such as the timing, frequency, scheduling, packet sizes and types (e.g., to emulate various services), location, etc. This enables the emulation of distinct scenarios to check if Quality of Service (QoS) or Service Level Agreement (SLA) are met. As a result, active monitoring permits to measure target metrics only when and where they are needed.

Because of the stateless forwarding performed by OpenFlow switches, as well as their incapability of generating and injecting any packets in the network, active measurements need to be entirely initiated and performed by the network controller. To that end, the logic of these measurements needs to be implemented as an application running on top of the controller. The following analyses the challenges of implementing current OAM procedures as applications on the network controller. Particularly attention is devoted to the Connectivity Check, Loopback, and Link Trace protocols as reference procedures because of their large deployment in today's operator networks.

Connectivity Check

This protocol requires the switch to generate, timestamp, and send specific messages over the data plane to measure the unidirectional bandwidth and delay of a link (see Figure 8.1(a)). However, OpenFlow does not provide any support for packet generation and injection. Therefore, the network controller needs to overcome such shortcoming and implement the CCM mechanisms as shown in Figure 8.1(a). Please note that the numbers (#) appearing in the following text refer to the distinct messages illustrated in Figure 8.1. The controller first generates and timestamps a CCM message for each CCM-enabled switch port and then forwards it to the switch over the control plane (A.1). Next, the switch forwards the CCM message over the data plane (A.2). The receiving switch finally notifies the network controller of the successful CCM reception by forwarding on the control plane the frame received over the data plane (A.3).

In this way, the network controller supervises the connectivity status by keeping track of the CCM messages being sent and received. However, the measurement of the one-way delay is inaccurate because (*i*) the timestamping occurs before the packets are actually transmitted over the data plane, and (*ii*) the comparison between the transmission and reception time is done in the controller and not on the target switch. Moreover, the measurement of the one-way bandwidth is inaccurate because the control plane bandwidth becomes a limiting factor for the data plane bandwidth. Clearly, such approach unnecessarily overloads the controller and the control plane. Indeed, adopting the most stringent connectivity checking configuration, that is to transmit a CCM message every 3.3 ms [ITU15c] on every port, generates a total of ∼340 Gbps over the control plane from the antenna sites to M4 nodes (see Figure 7.1), and several Tbps when considering 12 of those segments.

Loopback

This protocol requires the switch to generate, timestamp, and send specific messages over the data plane to measure the bidirectional bandwidth and delay of a link (see Figure 8.1(b)). The network controller needs to create the Loopback Message (LBM), timestamp it, and to send it over the control plane to the switch (A.1), which in turn forwards the message on the data plane (A.2). This is then intercepted and sent back to the controller (A.3), thus triggering a timestamped Loopback Reply (LBR). This is sent to the switch (B.1) and then forwarded on the data plane (B.2). The switch originating the initial message receives the Reply frame and forwards it back to the network controller (B.3). SDN-based implementation of Loopback message suffers from the same limitations as the Connectivity Check, both in terms of accuracy and overload of the control plane.

Link Trace

enables the hop-by-hop tracking of a certain path by sending a series of Linktrace Message (LTM) with incremental Time to Live (TTL) values[11] as shown in Figure 8.1(c). The network controller generates a LTM message ($TTL = 2$) and sends it over the control plane to the switch (A.1). Such message is then forwarded over the data plane (A.2) and back to the controller (A.3). Next, the controller generates a Linktrace Reply (LTR) with a TTL value equal to $TTL(LTM) - 1$ as response (B.1). Simultaneously, the controller decreases the TTL of the LTM message[12] and sends it back to the data plane (C.1). LTM and LTR messages are then forwarded on the data plane (B.2, C.2) and to the controller (B.3, C.3). Upon (C.3) reception, the controller generates an LTR (D.1) which is then transmitted on the data plane (D.2, D.3) and finally back to the controller (D.4). Link Trace potentially presents the same scalability issues[13] of Connectivity Check and Loopback. However, Link Trace is used for verifying the correct configuration of network paths and it is activated on-demand, presenting different scales of operation compared to CCM and LBM.

8.2.2 Towards a stateful OpenFlow?

The several issues of OpenFlow in performing active monitoring can be traced down to two factors: (*i*) the incapability of OpenFlow to keep information on the forwarded traffic, and (*ii*) the impossibility of generating and injecting packets. In the recent years several works have been proposed, fostering the debate on *stateless* vs *stateful* OpenFlow. Bianchi et al. [Bia+14][14] propose OpenState, an OpenFlow-compatible abstraction to formally describe stateful processing of flows inside the switch itself. Such abstraction relies on eXtended Finite State Machines and an Application Programming Interface (API) which can be implemented on the switch by largely

[11] An LTM with a TTL equal to 0 is discarded by the network. A LTM message with $TTL = n$ serves at determining the n^{th} hop.

[12] OpenFlow does not support CFM headers, thus the TTL cannot be decreased by the switch itself as e.g. in the IP protocol.

[13] Accuracy issues are not present because packets are not timestamped.

[14] G. Bianchi et al. 'OpenState: Programming Platform-independent Stateful Openflow Applications Inside the Switch'. In: *SIGCOMM Comput. Commun. Rev.* 44.2 (April 2014), pages 44–51. ISSN: 0146-4833. DOI: 10.1145/2602204.2602211.

reusing existing OpenFlow features. OpenState allows implementing reactive applications on the switches, such as port knocking which is commonly used for opening a port on a firewall. While this capability would be enough to keep track of incoming CCM messages for connectivity check, OpenState does not provide any support for packet generation on the switch.

Moshref et al. [Mos+14][15] propose FAST, which enables the controller to pre-install a state machine on the switch, thus allowing the switch to automatically record flow state transitions by matching incoming packets to installed filters. FAST defines an abstraction for state machines, a compiler for translating the state machines to the data plane API, and a data plane that includes a pipeline to support state machines with commodity switch components. Similarly to OpenState, FAST does not provide any support for packet generation.

Bifulco et al. [Bif+16][16] address such shortcoming by proposing an API that enables programmers to define in-switch packet generation operations, which include the specification of triggering conditions, packet's content and forwarding actions. The authors provide application examples for the delegation and implementation of Address Resolution Protocol (ARP) and Internet Control Messagge Protocol (ICMP) handling from the controller to the switch. However, the proposed API can trigger the packet generation only in reaction to the reception of a packet at the switch and not as reaction of some timed events. As a result, the proposed approach would be sufficient to generate Loopback Reply messages but not for the periodical generation of CCM messages.

Cascone et al. [Cas+][17] propose SPIDER, a packet processing pipeline design for stateful SDN data plane that allows the implementation of failure recovery policies with fully programmable detection and rerouteing mechanisms directly in the switches' fast-path. While SPIDER provides an OpenFlow-compatible way for fast-rerouteing based on heartbeat messages (like CCM), other monitoring features are not considered. For instance, link degradation and delay can not be measured in SPIDER, thus limiting the rerouteing to hard-connectivity failures only, that is no messages are received. Therefore, rerouteing based on the link delay or degradation is not possible. Moreover, SPIDER does not support the periodical reporting of the links status of (e.g., quality) to the controller.

Summarising, the works available in the literature propose different solutions for enabling stateful processing in the data plane mainly tailored to user traffic. However, active monitoring only requires a stateful processing of the packets involved in some measurements (e.g., CCM). The amount of such traffic is expected to be negligible compared to the user traffic. As a result, only a little portion of the traffic needs to be actively processed by the switch for telemetry purposes. For that reason, we advocate that extending OpenFlow processing from stateless to stateful processing for active monitoring purposes is not ideal. Indeed, such approach would bring considerably large complexity in the switch fabric compared to the small amount of monitoring traffic that requires stateful processing. Therefore, a lighter solution is required in terms of switch complexity. Nowadays OpenFlow switches are commonly equipped with general-purpose processors[18] which mainly interact with the switch fabric only for configuration and management purposes. In our view, such processors could be leveraged as well to implement the stateful processing required for the telemetry procedures. In this way, OpenFlow performs the stateless processing of user traffic as usual while monitoring traffic is processed locally on the CPU switch. This would allow to stay compatible with the current OpenFlow solution as detailed in the next section.

[15] M. Moshref et al. 'Flow-level State Transition As a New Switch Primitive for SDN'. in: *SIGCOMM Comput. Commun. Rev.* 44.4 (August 2014), pages 377–378. ISSN: 0146-4833. DOI: 10.1145/2740070.2631439.

[16] R. Bifulco et al. 'Improving SDN with InSPired Switches'. In: *Proceedings of the Symposium on SDN Research*. SOSR '16. Santa Clara, CA, USA: ACM, 2016, 11:1–11:12. ISBN: 978-1-4503-4211-7. DOI: 10.1145/2890955.2890962.

[17] C. Cascone et al. 'Fast failure detection and recovery in SDN with stateful data plane'. In: *International Journal of Network Management* 27.2 (), e1957. DOI: 10.1002/nem.1957.

[18] For instance, the NoviSwitch 21100 is equipped with an Intel Core i7 and the Advantech ESP-9230 with an Intel Xeon.

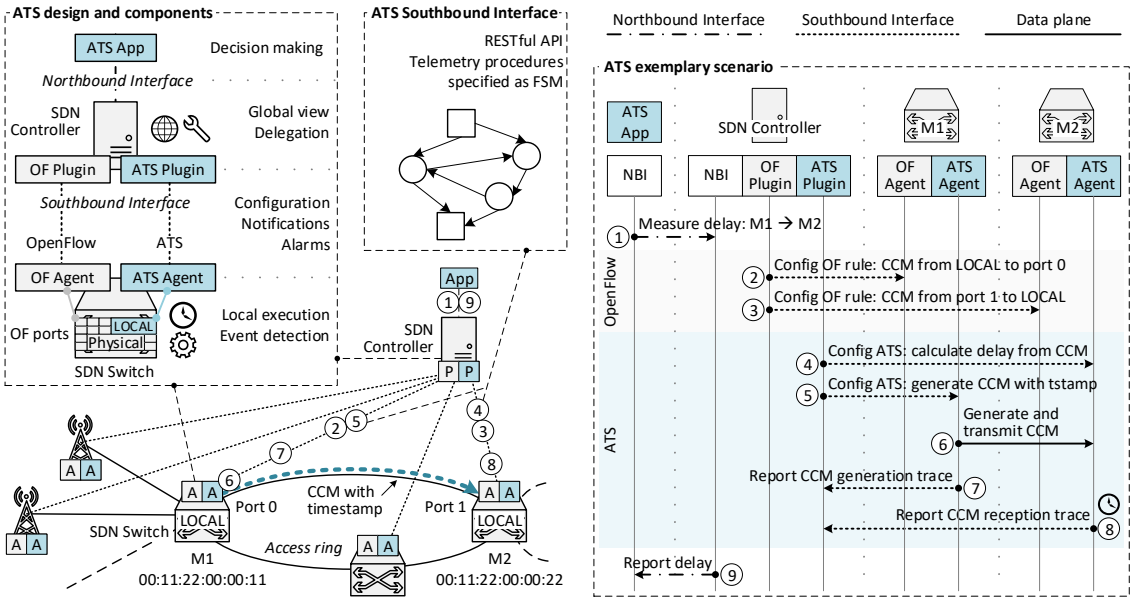


Figure 8.2: Adaptive Telemetry System components and exemplary scenario with message flow.

Table 8.1: Adaptive Telemetry System RESTful API.

URI	METHOD	DESCRIPTION
ATS plug-in on the network controller		
/ats/	GET	Information about the ATS plug-in
/ats/report/	GET	Reports available on the network controller
/ats/report/id	GET, POST, PUT, DELETE	Read, create, update, delete a report on the network controller
ATS agent on the switch		
/ats/	GET	Information about the ATS agent
/ats/fsm/	GET	ATS procedures available on the switch
/ats/fsm/id	GET, POST,PUT, DELETE	Read, create, update, delete an ATS procedure on the switch
/ats/fsm/id/event	POST	Send events to the ATS procedure

8.3 Design of an Adaptive Telemetry System

This section presents the design of our Adaptive Telemetry System (ATS) for enabling stateful data plane processing tailored to active monitoring. Specifically, Adaptive Telemetry System (ATS) aims at providing operators with a set of SDN-compliant tools for defining and configuring telemetry procedures on the switches. While the state-of-the-art solutions add extra features directly into OpenFlow protocol, we adopt an hybrid approach where the telemetry system interacts with the legacy OpenFlow pipeline, i.e., no extension is proposed to the current OpenFlow specifications. Therefore, such hybrid approach does not envision any change on the switch backplane, which is the part internal to a switch implementing the OpenFlow pipeline and in charge of forwarding packets between ports. Figure 8.2 shows the ATS design which envisages three main components: (*i*) an ATS application, (*ii*) an ATS plug-in, and (*iii*) an ATS agent.

8.3.1 ATS application

The ATS application runs on the controller and it is in charge of taking the decision of what, when, and where to measure. Since it runs on the controller, the application has a global view on the status of the network, and based on the active traffic flows, path configuration requests, and offered network services, the application decides what the parameters to be monitored in the underlying

network are (e.g., delay, link quality, etc.), either periodically or on-demand. The active execution of those measurements is then delegated to the switches which follow the instructions received by the controller.

8.3.2 ATS plug-in

The ATS plug-in runs on the controller and it is in charge of implementing the communication with the switches via a southbound interface. This interface exposes a RESTful API which provides a uniform and predefined set of operations to allow the network controller to dynamically configure the telemetry procedures on the switches and to receive notifications and alarms. Table 8.1 reports the Uniform Resource Identifier (URI) exposed by (*i*) the network controller via the ATS plug-in and by (*ii*) the switch via the ATS agent (see next paragraph). Specifically, the configuration of the telemetry procedures is based on a Finite State Machine (FSM), which is then executed locally on the switch. The FSM representation is further detailed in Chapter 8.4.1.

8.3.3 ATS agent

The ATS agent runs on the switches and it is in charge of (*i*) locally executing the FSMs configured by the network controller and (*ii*) sending the appropriate notifications and alarms via the common API. As described in Chapter 2, in addition to physical ports, OpenFlow defines logical and reserved ports internal to the switch that can be used by external applications/components to interact with the OpenFlow pipeline. For instance, the reserved port CONTROLLER is used to send a packet from the switch to the network controller and vice versa. Similarly, the reserved port LOCAL enables components running on the switch to directly interact with the OpenFlow network. As a result, the ATS agent uses the LOCAL port to send/receive packets over/from the data plane through the standard OpenFlow pipeline.

8.4 Exemplary scenario

This section provides an example of ATS operation, i.e., measuring the one-way delay between two switches directly connected in an access ring[19]. The exemplary scenario, including the network topology based on Figure 7.1 and the high-level message flow, is reported in Figure 8.2. Please note that the numbers (#) appearing in the following text refer to the distinct steps illustrated in the right-hand side of Figure 8.2.

At some point in time, an ATS application running on the network controller decides that it needs to measure the one-way delay between the switch M1 and the switch M2. For instance, such decision could be made in response to a path configuration request received by the network controller for time sensitive traffic. Next, the ATS application selects the most appropriate measurement procedure (e.g., CCM with timestamp) and asks the network controller to perform such measurement via the northbound interface (1). In turn, the network controller configures the necessary OpenFlow rules for forwarding the traffic to and from the ATS agent (2),(3) for such type of traffic. Let's assume that the port 0 of M1 switch is directly connected to the port 1 of M2 switch. In this case, the OpenFlow pipeline of M1 switch is instructed to forward the CCM messages generated by the ATS agent over the port LOCAL to the port 0. Similarly, the M2 switch is configured to forward the CCM messages received over the port 1 to the port LOCAL for being processed by the ATS agent. Steps (2),(3) are standard OpenFlow operations.

Then, the network controller configures on the switches via the ATS plug-in the measurement procedures in form of finite state machines (see Chapter 8.4.1 for further details). Specifically, the CCM reception and delay calculation is configured on the M2 switch (4) and the CCM generation on the M1 switch (5). For instance, the M1 switch is configured to generate a total of 100 CCM messages with an interval of 10 ms between subsequent packets (6). After having transmitted all

[19] The clocks of two switches are assumed to be synchronised. See Chapter 8.6 for additional considerations on clock synchronisation.

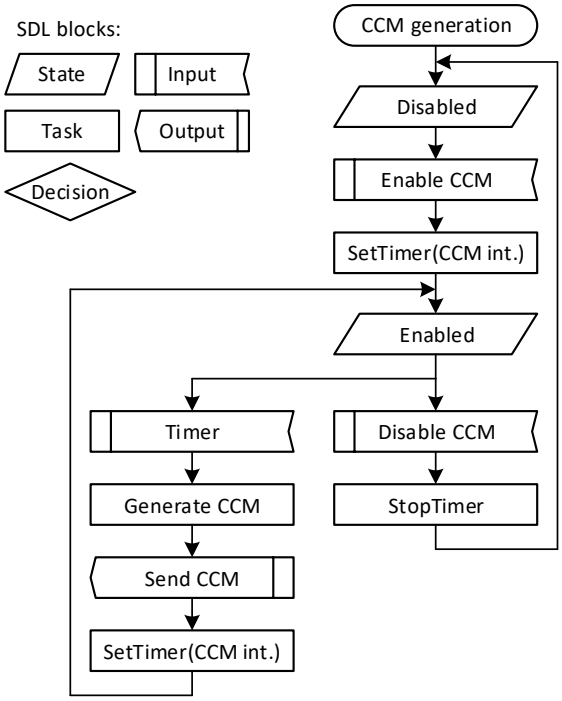


Figure 8.3: SDL Finite State Machine for CCM generation [ITU18a].

the CCM messages, the ATS agent is configured to report a trace of the generated messages (7). Similarly, the M2 switch is configured to compute the delay of each CCM message received and to report the trace of all computed values to the network controller once the last CCM is received (8). Additionally, a time-out is configured for dealing with the case of the last CCM going lost. Finally, the network controller informs the ATS application via the northbound interface on the measured delay (9).

8.4.1 ATS procedures modelling

The previous paragraphs briefly introduced that the telemetry procedures are specified via finite state machines. This decision is based on the analysis of the standard OAM operations as defined in ITU-T Y.1341 [ITU18a][20] (ITU-T Y.1371 corollary standard), which is reported in the following.

ITU-T Y.1341 formally describes the OAM procedures as finite state machines by using the Specification and Description Language (SDL) [ITU16c][21]. SDL is a language targeted at the unambiguous description of the behaviour of reactive and distributed systems. While SDL provides a rich set of functional blocks for behaviour description, only a limited set is required for fully describing the behaviour of the OAM procedures under consideration. Specifically, they can be described by exclusively using the following five SDL blocks:

1. *State*: describes the status of the FSM that is currently waiting to execute a transition. Two special states define the *entry* and *exit* points of the FSM respectively.
2. *Task*: defines a series of internal steps to be executed by the switch. Variable declaration and assignment, packet forging, and timer configuration are common tasks.
3. *Decision*: is equivalent to an *if-then-else* statement. Local variables, header fields of the incoming packets, and timestamps are the usual comparison terms.

[20] ITU-T. *Characteristics of Ethernet transport network equipment functional blocks*. Series G: Transmission Systems and Media, Digital Systems and Networks, Series Y: Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks G.8021/Y.1341. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), June 2018.

[21] ITU-T. *Operations, administration and maintenance (OAM) functions and mechanisms for Ethernet based networks*. Series Z: Languages and General Software Aspects for Telecommunication Systems Z.102. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), April 2016.

4. *Input*: is the actual trigger of the transition and an event is its common representation. The events leveraged in the OAM operations are: (*i*) incoming packet, (*ii*) external signal, and (*iii*) timer expiration.
5. *Output*: specifies a set of actions to be executed upon condition fulfilment or event reception. Packet transmission and signal firing are common outputs.

Figure 8.3 shows the SDL finite state machine for describing the CCM generation procedure at the switch [ITU18a]. While SDL can comprehensively describe the OAM protocols behaviour, it only provides a basic model for describing the supported data types (e.g., integer, real, char, etc.). A more comprehensive data model is hence required by the network controller to unequivocally instruct the switch. Therefore, we propose a comprehensive data model for telemetry procedures based on the combination of IETF RFC 7223, IEEE 802.1Qcp [IEE18a][22], and Metro Ethernet Forum (MEF) documents [MEF12a][23], [MEF12b][24]. By combining and extending them, our proposal takes the form of a YANG model specifying the telemetry procedures and the *port* and *packet* data types for enabling the generation, transmission and reception of packets, which are of paramount importance for the telemetry procedures.

With this data model, the next step is to design a generic FSM representation for telemetry procedures that can be exchanged between the controller and the switches. To that end, we define the following basic concepts for ATS FSM representation: *state*, *transition*, *event*, and *datamodel*. Each state contains a set of transitions that define how the FSM reacts to events, which can be generated by the state machine itself or by external entities (e.g., packet reception). The data model defines how the data internal to the state machine is stored, read, and modified as well as its interpretation in conditional expressions. In the following we report the main ATS elements expressed as Extensible Markup Language (XML) elements:

- <*state*>: this element holds the representation of a state and it can be used to express the SDL *State* block.
- <*data*>, <*assign*>: the <*data*> element is used to declare and populate portions of the data model whilst the <*assign*> element is used to modify the data entries. These ATS elements combined can represent the SDL *Task* block.
- <*if*>, <*elseif*>, <*else*>: allow to describe conditional code execution and consequently the SDL *Decision* block. Conditional expressions are supported on local variables as well as on header fields and timestamps.
- <*transition*>: defines the available transitions between states and the events that trigger them. The <*onexit*> and <*onenter*> elements are used to define whether the instructions need to be executed when leaving or entering a given state. Additionally, *Event I/O Processor* is used for emitting input and output events that result in state transition. To that end, a dedicated I/O processor is required to notify about incoming packets and trigger the transitions. This, along with *port* and *packet* data types, can be jointly used with the <*transition*> element to express the SDL *Input* block.
- <*send*>: this element is used to send events and data to external systems (e.g., to the network controller or to the data plane) and to raise events in the current system (e.g., raise a timer). This element can be used to express the SDL *Output* block and be leveraged, e.g., to fire an alarm from the switch to the network controller. Moreover, in conjunction with the *port* and *packet* data types, <*send*> can be used to transmit packets.

Code 8.1 and Code 8.2 show the ATS state machines implementing the one-way delay measurement described in Chapter 8.4. Particularly, Code 8.1 depicts the ATS state machine configured by the network controller on the M1 switch for generating CCM messages. This is the FSM sent by the network controller to the M1 switch in step (5) of Figure 8.2. Similarly, Code 8.2 presents

[22] IEEE. *Bridges and Bridged Networks Amendment: YANG Data Model*. Standards for Local and metropolitan area networks 802.1Qcp. Institute of Electrical and Electronics Engineers (IEEE), May 2018.

[23] MEF. *Service OAM Fault Management YANG Module*. Specification MEF 38. Metro Ethernet Forum, April 2012.

[24] MEF. *Service OAM Performance Monitoring YANG Module*. Specification MEF 39. Metro Ethernet Forum, April 2012.

the ATS state machine for computing the one delay based on received CCM messages on the M2 switch. This is the FSM sent by the network controller to the M2 switch in step (4) of Figure 8.2. As it can be noticed, the *<data>* element in the data model allows to define custom packets by concatenating multiple *<expr>* elements which represent the header and the payload structure. This is useful to create and manipulate packets to be transmitted (e.g., incrementing the sequence number) and to decode the received packets according to a defined structure. These packets can be then transmitted or received over the LOCAL port defined in the data model. Additionally, the reserved port *ctrl* is provided by the ATS agent to allow communication to and from the network controller (e.g., to send the delay report). Moreover, the ATS agent provides the data type *time* to access the clock on the switch (i.e., current time available via *time.now*). Finally, the Event I/O Process provides the event *pkt_in* to trigger a state transition when a packet is received.

```
<xml version="ats">
<datamodel>
 <data id="port" type="port" expr="local"/>
 <data id="ccm" type="packet">
  <expr id="ether_dst">00:11:22:00:00:22</expr>
  <expr id="ether_src">00:11:22:00:00:11</expr>
  <expr id="ether_type">89:02</expr>
  ... additional header fields ...
  <expr id="sn">00:00:00:00</expr>
  <expr id="timestamp">00:00:00:00</expr>
 </data>
 <data id="report" type="list" expr="[]"/>
</datamodel>
<state id="disabled">
 <onexit>
  <send event="timer" delay="0.01s"/>
 </onexit>
 <transition event="enable" target="enabled"/>
</state>
<state id="enabled">
 <onexit event="timer">
  <assign target="ccm.sn" expr="ccm.sn + 1"/>
  <assign target="ccm.timestamp" expr="time.now"/>
  <assign target="report" expr="report + [ccm.timestamp]"/>
  <send target="port" type="packet" data="ccm"/>
  <if expr="ccm.sn < 100">
   <send event="timer" delay="0.01s"/>
  </if><else>
   <send target="ctrl" type="list" data="report"/>
   <send event="disable"/>
  </else>
 </onexit>
 <transition event="timer" target="enabled"/>
 <transition event="disable" target="disabled"/>
</state>
</xml>
```

Code 8.1: ATS state machine for CCM generation.

```
<xml version="ats">
<datamodel>
 <data id="port" type="port" expr="local"/>
 <data id="ccm" type="packet">
  <expr id="ether_dst">00:00:00:00:00:00</expr>
  <expr id="ether_src">00:00:00:00:00:00</expr>
  <expr id="ether_type">00:00</expr>
  ... additional header fields ...
  <expr id="sn">0</expr>
```

```
10    <expr id="timestamp">0</expr>
11   </data>
12   <data id="report" type="list" expr="[]"/>
13  </datamodel>
14  <state id="disabled">
15   <onexit>
16    <send event="timeout" delay="1s"/>
17   </onexit>
18   <transition event="enable" target="enabled"/>
19  </state>
20  <state id="enabled">
21   <onenter event="pkt_in" type="packet" data="ccm">
22    <if expr="ccm.ether_src == 00:11:22:00:00:11 &&
23        ccm.ether_dst == 00:11:22:00:00:22">
24     <assign target="report" expr="report + [time.now - ccm.timestamp]"/>
25     <send target="port" type="packet" data="ccm"/>
26     <if expr="ccm.sn == 100">
27      <send target="ctrl" type="list" data="report"/>
28     </if>
29    </if>
30   </onenter>
31   <onexit event="timeout">
32    <send target="ctrl" type="list" data="report"/>
33    <send event="disable"/>
34   </onexit>
35   <transition event="pkt_in" target="enabled"/>
36   <transition event="disable" target="disabled"/>
37   <transition event="timeout" target="disabled"/>
38  </state>
39  </xml>
```

Code 8.2: ATS state machine for CCM reception.

8.5 Experimental evaluation

This section presents the experimental evaluation of ATS performance against legacy OpenFlow-based implementations. Figure 8.4 shows an overview of the test-bed used, comprising two machines equipped with an Intel Xeon E5-2620 processor, 128GB of RAM, and running Ubuntu 18.04 Server. One machine is used as network controller while the other is used to emulate the topology of one access ring (see Figure 7.1), which comprises one M2 node, six M1 nodes, and thirty-six antenna sites. The M1 nodes are assumed to be configured in a ring topology only at optical level. At logical level instead, they are connected point-to-point to their corresponding gateway (M2 node). This means that packets are enqueued only at gateway level, thus forming a logical tree topology (see Figure 8.4), which comprises a total of 43 nodes and 84 ports. Each node is then implemented as a LXD[25] container, which runs inside the ATS agent and Open vSwitch[26] as OpenFlow switch implementation.

The legacy-SDN implementation of the OAM protocols (see Chapter 8.2.1) is based on the OpenFlow controller Ryu[27]. Regarding the ATS implementation, the ATS agent translates the XML-based finite state machine into a JavaScript representation. We then used SCION-CORE[28] as interpreter for these procedures, which are executed on nodejs[29], a lightweight event-driven

[25] LXD: `https://linuxcontainers.org/lxd/introduction/`
[26] Open vSwitch: `https://www.openvswitch.org/`
[27] Ryu: `https://osrg.github.io/ryu/`
[28] Scion-core: `https://github.com/jbeard4/SCION-CORE`
[29] Nodejs: `https://nodejs.org/`

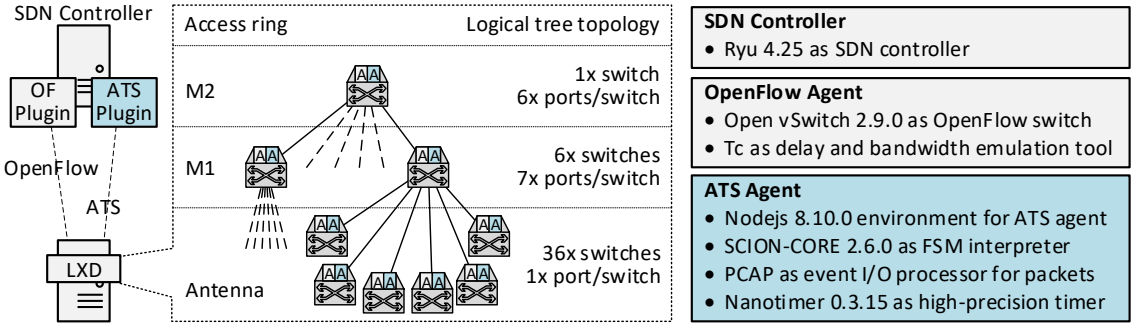


Figure 8.4: Test-bed overview and components.

environment. PCAP[30] and nanotimer[31] nodejs modules are used as packet-event I/O processor and high-precision timer, respectively. The results reported in the following are obtained by averaging 100 runs of each experiment.

8.5.1 Delay measurement

The first objective is to evaluate the legacy-SDN and ATS accuracy in measuring the one-way and two-way link delay. To that end, we implemented the CCM and LBM mechanisms on both systems and compared them against the baseline measurement obtained with the *ping* tool. It is worth highlighting that the ping is a network layer mechanism (i.e., IP) that is not usually available on the traditional switches operating at data-link layer (e.g., Ethernet). Nevertheless, such tool is available on our test-bed because the networks nodes are implemented on LXD, which provides a full-fledged operating system environment. Moreover, we used tc[32], which is a traffic control tool for Linux systems, to configure a fixed delay of 1 ms on the virtual links connecting the various switch instances.

The delay was then measured for an increasing number of simultaneously active ports and message generation interval. While the number of ports provides an estimation of how well the network controller scales with respect to the number of switches under its management, the message generation interval provides an estimation of how well the network controller scales with respect to the freshness of the measurements (e.g., the values are updated every 100 ms). To assess the first scalability aspect, we gradually activated a growing number of ports, starting from 1, with an incremental step of 4 ports until reaching 84 ports being simultaneously active. That is, we tested the systems under the scenarios of 1 active port, 4 ports, 8 ports, etc., until 84 ports. Regarding the second scalability aspect, we selected the three highest transmission rates defined in [ITU15c], which result in a generation intervals of 3.3 ms, 10 ms, 100 ms, respectively (see Chapter 8.2.1). Finally, we performed 100 runs for each of the scenarios obtained by combining the number of active ports and the message interval.

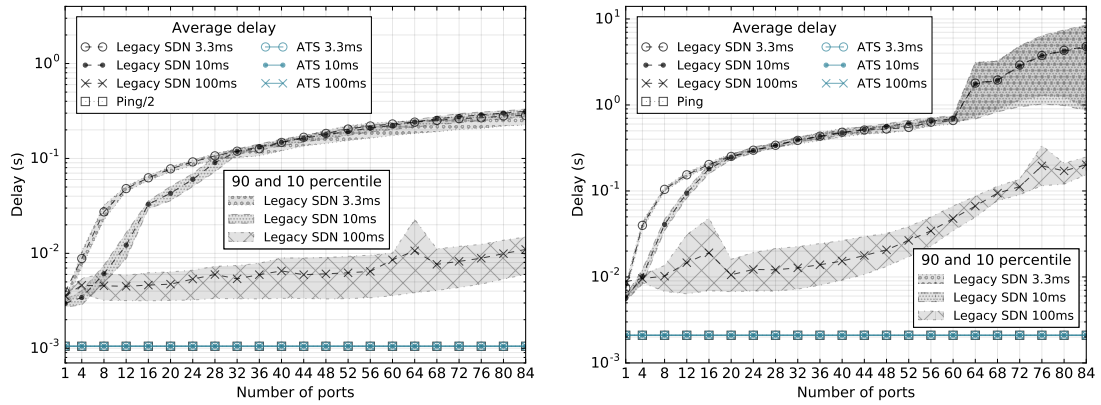
Figure 8.5(a) and Figure 8.5(b) show the results for the one-way and two-way delay measurement, respectively. Noticeably, Figure 8.5(b) shows that the round-trip delays measured via ATS and ping are comparable and they do not depend on the number of active ports nor on the message interval. This is also highlighted in Table 8.2 which reports the statistical characteristics of the delay measurements. More precisely, ping reports an average two-way delay of 2.10 ms which matches the value reported by our LBM implementation on ATS. For what concerns the one-way delay, we consider ping/2 as baseline since our test-bed is characterised by symmetric links. Moreover, since all the switches are running on the same physical machine, we can safely compute the one-way delay with CCM messages because all the containers share the same CPU clock (see Chapter 8.6 for additional considerations). According to these considerations, the one-way delay obtained with ping/2 is 1.05 ms. As expected, the CCM implementation on ATS reports an average one-way delay

[30] Node-pcap: `https://github.com/node-pcap/node_pcap`
[31] Nanotimer: `https://github.com/Krb686/nanotimer`
[32] Tc: `http://tldp.org/HOWTO/Traffic-Control-HOWTO/intro.html`

Table 8.2: Statistical characteristics of the delay in milliseconds measured with ping, legacy-SDN controller, and ATS.

	SOLUTION	#PORTS	INTVL	MEAN	5TH	95TH	MODE	MED	STD
(1)	Ping/2	any	any	1.05	1.02	1.08	1.05	1.05	0.02
	ATS	any	any	1.05	1.01	1.10	1.05	1.05	0.03
	SDN	1	10	2.96	2.66	3.46	2.81	2.90	0.37
	SDN	84	10	313.77	260.98	531.49	298.43	295.40	88.23
(2)	Ping	any	any	2.10	2.04	2.16	2.10	2.10	0.04
	ATS	any	any	2.10	2.06	2.14	2.06	2.10	0.03
	SDN	1	10	5.74	5.10	6.36	5.44	5.48	0.43
	SDN	84	3.3	4679.80	667.61	8852.62	622.69	4806.67	2640.45

(1) One-way delay (CCM); (2) Two-way delay (LBM);



(a) One-way delay measurement with CCM messages. (b) Two-way delay measurement with LBM messages.

Figure 8.5: Scalability of one-way and two-way delay measurement with ping, legacy-SDN controller, and ATS.

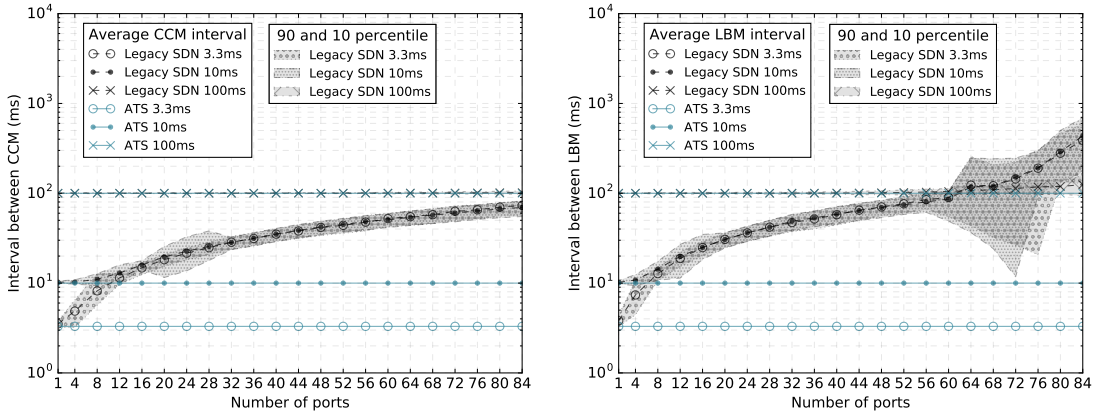
of 1.05 ms, matching the ping/2 value. Therefore, ping and our ATS implementation provide similar accuracy in measuring the one-way and two-way delay. It is worth highlighting that Figure 8.5(a) and Figure 8.5(b) do not show the confidence intervals for the ping and ATS data because they are not graphically appreciable.

Regarding the legacy-SDN solution, the measured delay depends on the number of active ports and on the configured message interval. More precisely, the delay measurement closest to ping and ATS occurs in case of 1 active port and a message interval of 10 ms for both CCM and LBM messages. In case of CCM, the network controller reports an average of one-way delay of 2.96 ms. In case of LBM instead, the network controller reports an average of two-way delay of 5.74 ms. In both cases, the reported value is $\sim$ 300% higher than the delay measured by ATS and ping because every message sent over the data plane requires two additional messages on the control plane. As it can be evinced in Figure 8.5(a) and Figure 8.5(b), the delay measured by the network controller significantly increases with the number of ports and with smaller message generation intervals. The highest values for the one-way delay is obtained in case of 84 ports and a message interval of 10 ms (similar results are obtained in case of 3.3 ms). The average delay with CCM is 313.77 ms. Similarly, the highest values for the two-way delay is obtained in case of 84 ports and a message interval of 3.3 ms with an average measured delay of 4679.80 ms. It is clear that the delay measured by the network controller is far from the reality being several orders of magnitude larger than the reference value.

By comparing the above results with the performance requirements for low-latency and high-reliability scenarios defined by 3GPP [3GP18i], we can see that measuring the delay with legacy-

Table 8.3: Statistical characteristics of the message generation interval in milliseconds with legacy-SDN controller and ATS.

SOLUTION		#PORTS	INTVL	MEAN	5TH	95TH	MODE	MED	STD
ATS	CCM/LBM	any	3.3	3.30	3.27	3.33	3.30	3.30	0.16
	CCM/LBM	any	10	10.00	9.97	10.03	10.00	10.00	0.12
	CCM/LBM	any	100	100.00	99.96	100.04	100.00	100.00	0.24
SDN	CCM/LBM	1	3.3	3.52	3.13	3.96	3.47	3.50	0.88
	CCM/LBM	1	10	10.24	9.73	10.80	10.24	10.23	0.44
	CCM/LBM	1	100	100.33	99.70	101.01	100.46	100.31	2.14
	CCM	84	3.3	73.85	54.79	85.79	72.59	69.47	58.17
	LBM	84	3.3	440.20	48.94	976.95	449.40	395.59	341.98



(a) Average time gap between two subsequent CCM messages.

(b) Average time gap between two subsequent LBM messages.

Figure 8.6: Scalability of message generation for legacy-SDN controller and ATS.

SDN does not provide the necessary accuracy for critical services. For instance, 3GPP defines that *discrete automation* traffic requires a maximum end-to-end latency of 10 ms and a jitter of 100 μs. *Electricity distribution* instead requires an end-to-end latency of 5 ms and a jitter of 1 ms. Even in the most favourable case in legacy-SDN of measuring the one-way delay on 1 port at a time, the jitter on a single link is reported as 0.37 ms, which is above the maximum admissible value for *discrete automation*. Similarly, with 4 simultaneously active ports in legacy-SDN, the one-way measured delay is 4.61 ms with a jitter of 0.77 ms. This makes difficult to assess, i.e., whether the 5 ms end-to-end requirement is met for the *electricity distribution* service. On the contrary, with our implementation of ATS we can safely measure the delay with a very limited jitter (e.g., 30 μs in case of CCM over one link). By reporting the measurements obtained by ATS, the network controller can hence safely decide whether a link is suitable for a given set of services, even the most stringent ones requiring a maximum jitter of 100 μs.

8.5.2 Connectivity status

The second objective is to evaluate how effectively the CCM and LBM messages can be used to detect the link status. Also in this case the evaluation was performed for an increasing number of simultaneously active ports (i.e., from 4 to 84 with a step of 4) and message generation interval (i.e., 3.3 ms, 10 ms, 100 ms). As it can be evinced from the previous evaluation, the overload suffered by the legacy-SDN controller produces an over-estimation of the link delay as a side effect. Since the network controller is not capable of generating and processing the CCM and LBM messages in time for all the ports, the network controller starts queuing the messages. This produces a gradual increment of the time gap between two subsequent messages, thus deviating

from the configured interval. Figure 8.6(a) and Figure 8.6(b) highlight the diverging trend for CCM and LBM, respectively, by depicting the time difference measured between subsequent messages for an increasing number of ports and different message generation intervals. The results show that ATS is capable of generating the messages in compliance with the configured interval regardless the protocol and the number of active ports as reported in Table 8.3.

Regarding the legacy-SDN solution, in case of 1 active port, the network controller generates CCM and LBM messages with an interval quite close to the target one as shown in Table 8.3. In case of 84 ports instead, the average message interval significantly diverges from the target one. For instance, CCM messages are generated with an average interval of 73.85 ms for the 3.3 ms case, 71.13 ms for the 10 ms case, and 101.62 ms for the 100 ms case. LBM messages instead are generated with an average interval of 440.20 ms for the 3.3 ms case, 426.76 ms for the 10 ms case, and 124.54 ms for the 100 ms case. Additional statistical characteristics are reported in Table 8.3 for the extreme cases of CCM and LBM generated every 3.3 ms on 84 ports.

By comparing the above results with [3GP18i], we can see that supervising the connectivity status with legacy-SDN does not provide the necessary responsiveness for critical services. For instance, the *electricity distribution* the *process automation* services are characterised by a survival time of 10 ms and 100 ms, respectively. The survival time indicates the admissible maximum time for restoring the connectivity in case of a link failure or in case the delay requirement is no longer met (see Chapter 8.5.1). According to the CCM protocol [ITU15c], a connectivity failure is detected if no heartbeat messages are received within 3.5 times the configured interval (e.g., 11.55 ms in 3.3 ms case). Given the precision of ATS in periodically generating the messages, it is easy to detect whether no heartbeat messages are received before the time-out expiration. However, this is not the case for legacy-SDN since the increasing gap between two subsequent messages yields to a considerable amount of false-positive failure detections. For instance, in the extreme case of 84 ports and 3.3 ms target interval, CCM messages are generated every 73.85 ms, which is considerable higher than the time-out of 11.55 ms for detecting a link failure. In our set-up, such issue is not present for the 100 ms case, thus allowing legacy-SDN to detect a connectivity failure within 350 ms. Nonetheless, this provides a measurement granularity of 100 ms which is not sufficient to comply with the above survival time requirements.

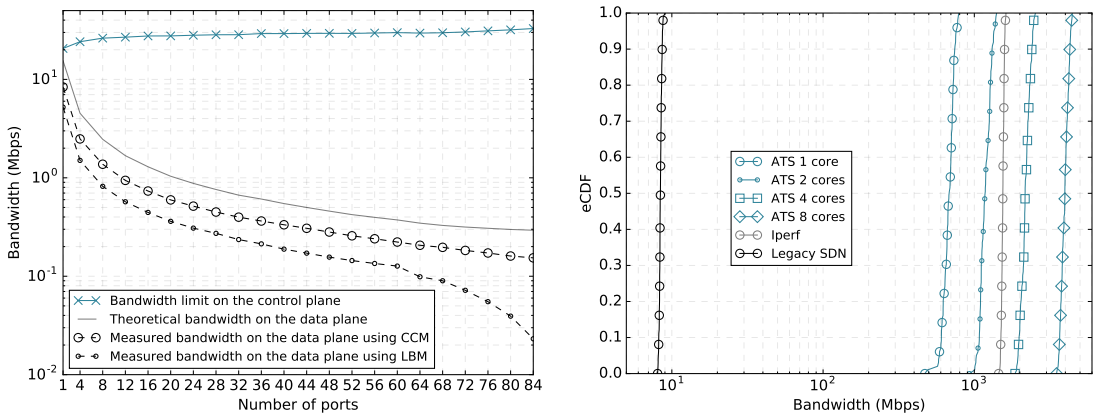
As it can be noticed, the survival time for the *electricity distribution* flow is 10 ms, which is smaller than the time-out of 11.55 ms when generating messages every 3.3 ms. Therefore, a smaller message interval is required to comply with that requirement. While this is easily achievable with ATS by simply updating the ATS state machine, it is undeniably harder in the legacy-SDN solution because of the scalability issues already appreciable with higher intervals. Moreover, the control plane delay may be taken into account to select the most appropriate message interval for the data plane. For example, in case of a control plane delay of 2 ms, a message interval of 1 ms would allow the network controller to receive a notification (e.g., link failure, maximum delay exceeded, etc.) within 5.5 ms, thus leaving 4.5 ms to restore the connectivity in case of a survival time of 10 ms. Finally, with this notification mechanism, ATS allows to offload the control plane by only transmitting information when certain conditions occur in the data plane.

8.5.3 Bandwidth measurement

The last objective is to evaluate the legacy-SDN and ATS accuracy in measuring the bandwidth available on a link. Similar to the previous cases, the evaluation was performed for an increasing number of simultaneously active ports (i.e., from 4 to 84 with a step of 4). In this case, messages are generated as fast as possible in order to saturate the available bandwidth. Figure 8.7(a) shows the control plane load and the bandwidth measured on the data plane by the network controller in case of legacy-SDN solution. Notably, the control plane load remains constant regardless the protocol (i.e., CCM or LBM) and the number of active ports. This is because the network controller is capable of generating only a fixed amount of packets per seconds. Particularly, the network controller generates an average of 708 packets per second, which result in an average control plane

Table 8.4: Statistical characteristics of the bandwidth in Mbps measured with Iperf, legacy-SDN controller, and ATS with 1 active port.

SOLUTION		#CPU	MEAN	5TH	95TH	MODE	MED	STD
Iperf	UDP	1	1543	1487	1595	1526	1544	33
ATS	CCM	1	677	587	758	622	682	59
	CCM	2	1186	1032	1347	1210	1178	99
	CCM	4	2186	1946	2434	2168	2167	156
	CCM	8	3970	3614	4334	3955	3499	234
SDN	CCM	1	8.25	4.72	12.33	8.45	7.69	2.87



(a) Legacy-SDN data plane bandwidth measurement and control plane load.

(b) One-way bandwidth measurement with CCM messages with increasing number of CPU cores.

Figure 8.7: Bandwidth measurement with legacy-SDN controller, ATS, and Iperf.

load of 12.21 Mbps.

Because of the overhead introduced by the encapsulation of CCM and LBM messages in OpenFlow messages between the network controller and the switches, only an average of 8.25 Mbps is then forwarded on the data plane. Since the number of generated packets is constant, these are spread over all the active ports resulting in a measured bandwidth inversely proportional to the number of ports as shown in Figure 8.7(a). In the case of 84 ports, the average bandwidth measured per port is 0.146 Mbps with CCM, while it is 0.022 Mbps with LBM. As it can be noticed, the bandwidth values provided by CCM are higher than LBM because CCM involves the generation of fewer packets compared to LBM (see Figure 8.1(a) and Figure 8.1(b)). This is further highlighted in case of LBM by the network controller saturation starting with 64 ports.

Moreover, we compare the obtained measurements with the maximum theoretical bandwidth measurable on the data plane by carrying the CCM and LBM messages as a payload over the TCP-based OpenFlow control channel. The achievable throughput for data transmitted over TCP is $\sim 75\%$ of the available link bandwidth[33]. In our scenario, the control plane bandwidth is shared among all the connected switches. Figure 8.7(a)) shows that the experimental results follow the same trend as the theoretical value (grey line). This implies that even if the network controller were capable of generating messages at the desired rate, the bandwidth measurement would always be distorted by the TCP connection used in the OpenFlow control channel. Figure 8.7(b) shows the one-way bandwidth measurement over one link performed with legacy-SDN, ATS, and iperf[34], which is our reference tool for active measurements of the available bandwidth on a link. We configured iperf to generate UDP packets to be comparable with CCM where messages are not

[33] TCP/IP performance factors: `https://www.netcraftsmen.com/tcpip-performance-factors/`
[34] Iperf: `https://iperf.fr/`

acknowledged. On our platform, iperf is capable of generating an average of 1.543 Gbps, while ATS running on a single CPU core is capable of generating an average of 0.677 Gbps. By increasing the number of CPU cores simultaneously generating the messages, ATS linearly increases the measured bandwidth. This is because each CPU core is capable of generating an average of $\sim$ 50 000 packets per second on our system.

By comparing the above results with the performance requirements for high data rate and traffic density scenarios defined by 3GPP [3GP18i], we can see that legacy-SDN is not capable of measuring whether there is enough bandwidth even for the least demanding service (15 Mbps of experienced data rate). On the contrary, ATS is capable of generating more than 1 Gbps with 2 CPU cores, which is the expected data rate experienced per user in indoor scenarios. Such measurements are expected to be performed on-demand upon a path configuration request to verify the fulfilment of the bandwidth SLA. Finally, it is worth highlighting that our ATS implementation is based on JavaScript for prototyping reasons, while iperf is written in C, a language that provides considerably higher performance. Even though we matched and surpassed the performance of iperf in generating traffic, this came at the cost of using more CPU cores.

8.6 Implementation and deployment considerations

In addition to the comparative tests previously described, we performed some experiments to obtain a deeper insight on ATS, especially in the CCM case. Special attention was paid to the CPU load and to the scalability with regard to the total number of ports. Particularly, we addressed the periodic generation of messages over multiple ports which was causing a computational outage on the legacy-SDN controller. To avoid such issue in ATS, we opted for using a packet template stored in a template buffer associated to each port. Such approach allows to pre-load a template of the message on each port and to trigger its transmission every interval. Each transmission only requires the modification of few bytes in the buffer (e.g., sequence number) thus reducing the total number of instructions to be executed. We tested our ATS implementation with a CCM interval of 3.3 ms on an emulated switch comprising 256 ports and we observed that the CPU load was (*i*) mainly due to the interrupts generated by the high-precision timer, and (*ii*) almost independent of the number of active ports. As a result, our implementation was able to transmit a CCM message every 3.5 ms on each of the 256 ports whilst running on a single core.

Finally, time synchronisation between the switches is required to measure i.e. the one-way delay and to have a common reference time for monitoring. In carrier grade networks there are two widely-adopted options for distributing the clock (a.k.a. frequency synchronisation): IEEE 1588 [IEE08] and Synchronous Ethernet [ITU15d][35]. The former defines a cost-effective packet-based clock distribution mechanism capable of providing a timestamp resolution of 8 ns with an accuracy 25 ns. The latter, instead, incorporates in the clock signal in the Ethernet physical layer, that is no ad-hoc messages for synchronisation are sent, and it is capable of providing sub-nanosecond accuracy. While the former option still provides a good accuracy for monitoring whilst being cheaper than the latter, it may occur that the clock distribution messages interfere with the network measurements and vice-versa. Therefore, it is important to configure appropriate QoS policies on the switches so as to avoid the disruption of the clock distribution eventually caused by the network measurements.

[35] ITU-T. *Timing characteristics of a synchronous Ethernet equipment slave clock*. Series G: Transmission Systems and Media, Digital Systems and Networks, Series Y: Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks G.8262/Y.1362. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), January 2015.

9. Conclusions of Part Two

The second part of this thesis has analysed and designed a Software Defined Networking (SDN)-based unified data plane architecture for 5G, namely crosshaul, based on two main components: (*i*) the Crosshaul Forwarding Element (XFE) and (*ii*) the Crosshaul Common Frame (XCF). The XFE is a multi-layer switch based on packet – Crosshaul Packet Forwarding Element (XPFE) – and circuit – Crosshaul Circuit Switching Element (XCSE) – switching elements. Unified forwarding is enabled by the XCF format that is common across the various types of traffic and the various link technologies in the network. As a consequence, the unified data plane enables a common management of the integrated network in a SDN fashion. Therefore, traffic requirements, and hence services, could be easily enforced onto the network by leveraging the integrated and harmonised view provided by the unified data plane. As a result, the network operational costs can be significantly reduced.

> Publications covering the design of the crosshaul network and related concepts are [Cav+17], [Dei+17], [Dei+16], [L C+16], [L C+18e], [Li+17].

> Patents covering the design of the crosshaul network and related concepts are [LMK18], [L C+17b], [L C+17c].

> The standard contribution [Ber+16] covers the crosshaul network requirements and related concepts.

Moreover, this part presented a characterisation of a 5G transport network and the expected traffic mixture of network slices. Several simulations have been performed to understand the role of queueing disciplines in different scenarios, such as urban, industrial, and rural. The results have been compared with the constraints of the traffic flows defined in 3rd Generation Partnership Project (3GPP) and criticality has been identified for the motion control traffic part of the Ultra-Reliable and Low Latency Communications (URLLC) slice. Jitter requirements for such flow are only satisfied when the traffic is terminated in the access ring and a strict priority with preemption queueing discipline is used. Regarding the other flows and slices, traffic requirements are fulfilled in a failure-free scenario where the protection ring in the access and aggregation is not activated.

> Publications covering the characterisation of a crosshaul network and related concepts are [L C+18e], [Mar+18].

> The open source simulator developed for characterising the crosshaul network has been published with the name of SimPype and it is available at: `https://simpype.readthedocs.io/en/latest/`.

Furthermore, this part has identified a gap between current SDN solutions and carrier grade network requirements under Operations, Administration and Maintenance (OAM) point of view. An analysis of widely-deployed OAM and SDN technologies has been hence performed showing that the stateless nature of OpenFlow poses significant scalability and accuracy problems in monitoring and managing the network. To overcome these issues, this part proposed an Adaptive Telemetry System (ATS) to enable locally on the switches active measurements (e.g., delay, bandwidth, etc.) and their reporting (e.g., alarms). The design approach chosen for ATS showed to provide compatibility with standard OpenFlow switches and controllers. An Application Programming Interface (API) has been defined for enabling the remote configuration of telemetry procedures, which adopt a Finite State Machine (FSM) implementation. This enables the switches to locally execute the stateful procedures required for active monitoring. Finally, an experimental evaluation has been presented, showing the benefits of ATS compared to legacy-SDN solutions. Particularly, ATS proved to bring significant benefits in terms of offloading the control plane, and the Network Controller (NC), as well as higher accuracy in the performed measurements, which comply with the performance requirements defined by 3GPP for 5G networks.

The publication [L C+18a] covers the design of an Adaptive Telemetry System (ATS) for 5G SDN-based transport networks.

The patent [Per+17b] covers OAM functionalities in SDN-based networks employing the OpenFlow Southbound Interface (SBI).

While the first and second part of this thesis have analysed the SDN control and data planes with the goal of increasing network flexibility and reducing costs, a second promising approach exist aiming at the same goal: Network Function Virtualisation (NFV). Similarly to SDN, which aims at decoupling the control from the data plane, NFV decouples the network functionalities from the underlying hardware, thus enabling the virtualisation of the network. As a result, SDN and NFV are orthogonal paradigms whose combination and integration is acknowledged as the true enabler for network flexibility and cost reduction. By leveraging the virtualisation technologies, contents, services, and applications can be dynamically hosted closer to the users so as to offload the core network and reduce the communication delay, which is a critical aspect for URLLC. The next part of this thesis therefore focuses on the virtualisation and computing aspects which can be beneficial to network flexibility.

III Part Three

10. A novel approach for multi-access convergence

Apart from achieving higher data rate than its predecessor, 5G also aims to satisfy several other technical requirements in a bid to cope with various emerging applications [3GP16b]. Specific applications such as Augmented Reality (AR), connected vehicles, and robotics require reliable communications with very low end-to-end latency to deliver high quality services [Mon+17][1]. Fulfilling such requirements is extremely challenging for a centralised network architecture and requires the gradual shifting of networking, computing, and storage capabilities closer to the end users to eliminate the delay caused by data transfer to distant cloud servers. This approach integrates and extends the *edge* and *fog computing* approaches as explained in the following.

Nowadays, most end user devices may operate multiple independent Radio Access Technology (RAT) in parallel (e.g., Long Term Evolution (LTE) and Wi-Fi). Such diversity can be exploited for example for traffic offloading purposes. However, this requires harmonisation and/or integration of communication protocol stacks from different RATs [Pen+16][2], selection of the best RAT for a given user/service at a given time [Lag14][3], or interference minimisation of different RATs sharing the same spectrum [Wan+15b][4]. With the intelligent hand-off, a paradigm shift of multi-RAT convergence can be envisioned, wherein context information of different RATs could be leveraged jointly to enhance network performance, cost-effectiveness, and user Quality of Exprerience (QoE). Motivated by these needs, the European Telecommunications Standards Institute (ETSI) has been the first to address this need by providing the framework of Mobile Edge Computing (MEC) [ETS16a], which is supported by the concept of Network Function Virtualisation (NFV) that was also pioneered by ETSI [ETS16b]. ETSI has further re-branded MEC as *Multi-Access Edge Computing* to remark its goal of achieving multi-RAT coordination via the edge.

Recently, the concept of *fog computing* has emerged driven by the Internet of Things (IoT) due to the need of handling the data generated from the end-user devices close to the edge [CZ16][5]. The term *fog* is referred to as computational resources such as the network nodes (e.g., other devices and access points etc.) surrounding and including the end user devices and is regarded as an effective

[1] R. S. Montero et al. 'Extending the Cloud to the Network Edge'. In: *Computer* 50.4 (April 2017), pages 91–95. ISSN: 0018-9162. DOI: 10.1109/MC.2017.118.

[2] M. Peng et al. 'Fog-computing-based radio access networks: issues and challenges'. In: *IEEE Network* 30.4 (July 2016), pages 46–53. ISSN: 0890-8044. DOI: 10.1109/MNET.2016.7513863.

[3] X. Lagrange. 'Very tight coupling between LTE and Wi-Fi for advanced offloading procedures'. In: *2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. April 2014, pages 82–86. DOI: 10.1109/WCNCW.2014.6934865.

[4] H. Wang et al. 'SoftNet: A software defined decentralized mobile network architecture toward 5G'. in: *IEEE Network* 29.2 (March 2015), pages 16–22. ISSN: 0890-8044.

[5] M. Chiang and T. Zhang. 'Fog and IoT: An Overview of Research Opportunities'. In: *IEEE Internet of Things Journal* 3.6 (December 2016), pages 854–864. ISSN: 2327-4662. DOI: 10.1109/JIOT.2016.2584538.

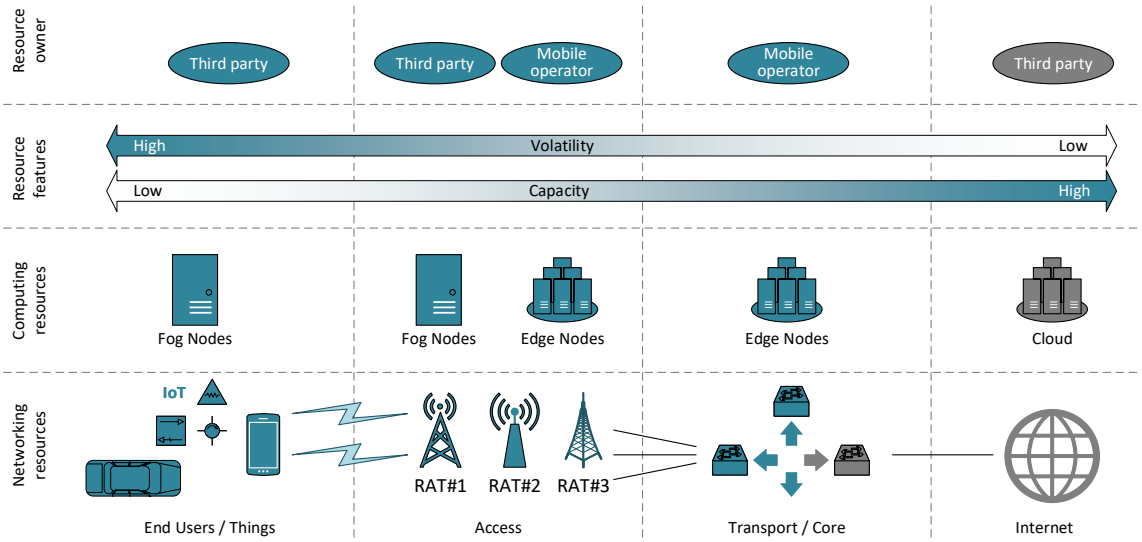


Figure 10.1: Edge and fog resources and characteristics.

approach for meeting requirements on latency, bandwidth, and computational resources [CZ16]. To that end, the industry-led OpenFog consortium was recently established with the aim of realising a cloud-to-thing continuum that distributes applications anywhere between cloud and things [Con17]. As a consequence, the edge is extended by amalgamating fog components. While the ETSI MEC approach provides computation capabilities near the end users via static substrates (e.g., data centres or servers) deployed at the edge, the harmonised edge and fog domain also encompasses and integrates the computational substrates on the move. For instance, computing, storage and network connectivity can be provided by any in-vehicle nodes or devices close to end users. Figure 10.1 shows the edge and fog resources (in blue) which may interact with centralised core and cloud domains (in grey) for offering a real cloud-to-thing continuum. By concentrating and sharing information extracted from multiple different RATs at this unified edge/fog logical platform, tight coordination among different RATs can be carried out using flexible and scalable computational resources depending on the context.

10.1 Towards a multi-domain environment

In addition to be physically distributed, the edge and fog resources may also belong to multiple stakeholders that cooperate with each other to achieve a capillary coverage of their services. Integrating resources belonging to distinct administrative domains is a challenge that goes beyond the pure technological dimension and involves trust relationships between parties. To that end, federation provides the means for integrating multiple administrative domains at different granularity into a unified platform where the federated resources can trust each other at a certain degree, whereas the federation trust is the embodiment of a service/business-level agreement or partnership between two organisations [Sim+16][6]. This trend is being embraced by today's cloud service providers in order to reach their sparse customers and fulfil their needs and concerns, such as customer's privacy and data ownership. A cloud federation can hence integrate a pool of diverse services from multiple service providers that self-govern each other by using well-defined interfaces and agreements between them [ABT14][7]. As a result, federation is a key enabling technology for cooperative service deployment in cloud environments. In a dynamic fashion, it allows heterogeneous and

[6] C. Simon et al. '5G exchange for inter-domain resource sharing'. In: *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. June 2016, pages 1–6. DOI: 10.1109/LANMAN.2016.7548842.

[7] M. R. M. Assis, L. F. Bittencourt and R. Tolosana-Calasanz. 'Cloud Federation: Characterisation and Conceptual Model'. In: *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*. UCC '14. IEEE Computer Society, 2014, pages 585–590. ISBN: 978-1-4799-7881-6. DOI: 10.1109/UCC.2014.90.

independently administrated clouds to interact and share resources with each other. Federated clouds offer an integrated cloud service by federating infrastructures provided by different cloud service providers. The ability of cloud federation to share cloud resources among participating service providers improves resource utilisation and enhances elasticity and reliability of cloud service. Federated clouds also enable new business opportunities. Virtualisation technologies and its orchestration, including the use of virtual machines and containers, play a major role in the provisioning of elastic mobile services in federated clouds. In this case, the federation mechanisms should include functionalities such as deployment, runtime management and monitoring, termination, authentication, access control and live migration of services in remote clouds [ABT14].

Many existing works in the literature develop frameworks and architectures to enable provisioning and management of services in federated clouds. Depending on the cooperation model of participants, cloud federation can be classified into several types. The first one is a horizontal federation, where participants cooperate on a peer-to-peer basis. This type of federation well applies to the case of federated mobile edge systems. The second type is a vertical federation, where participants are entities in a hierarchy, like hybrid cloud [Li+15][8], [LXX14][9] which combines the services provided by a private cloud and a third-party public cloud. This type of federated edge and fog architecture refers to the federation between edge and fog systems, between central cloud and edge system, or between central cloud and fog system. Finally, the third type of federation comprises both horizontal and vertical federations. Most existing federated clouds fall into the category of the vertical federation. For instance, Follow-Me Cloud (FMC) [TKF18][10] proposes an architecture for federated cloud and distributed mobile network environment which allows the services delivery through an optimal service anchor and the possibility of following mobile users as they roam through federated cloud environments. FMC utilises Markov-Decision-Process to make cost-effective and performance optimised migration decisions. Furthermore, challenges which cloud providers may face when participating in a federated cloud environment include the heterogeneity of cloud management systems and models describing the services. To resolve this issue, [Pan+17][11] proposes a coordinated application deployment system (CADS) to enable the description of the desired service deployment in form of a topology model. In this way, CADS provides interoperability in the deployment of services in federated clouds.

The NFV Industry Specification Group (ISG) defines a Management and Orchestration (MANO) framework [ETS16b] for deploying network services on an NFV environment. Nowadays, NFV MANO scope is limited to a single mobile operator network. To overcome such limitation, an NFV Work Item has been recently approved with the aim of enabling the management and orchestration across multiple operators [ETS18][12]. Although logical inter-connection between different mobile operators is being defined, integration with third-party domains (e.g., fog or cloud) is still not considered. Like NFV, ETSI MEC framework [ETS16a] only considers a single network operator domain and does not consider integration with third-party domains like fog. Finally, although ETSI MEC and NFV enable mobility of applications and services, it is only within the boundaries of the stationary edge resources of the mobile operator and volatile resources are not considered.

[8] J. Li et al. 'A Hybrid Cloud Approach for Secure Authorized Deduplication'. In: *IEEE Transactions on Parallel and Distributed Systems* 26.5 (May 2015), pages 1206–1216. ISSN: 1045-9219. DOI: 10.1109/TPDS.2014.2318320.

[9] Y. Lu, X. Xu and J. Xu. 'Development of a Hybrid Manufacturing Cloud'. In: *Journal of Manufacturing Systems* 33.4 (2014), pages 551–566. ISSN: 0278-6125. DOI: 10.1016/j.jmsy.2014.05.003.

[10] T. Taleb, A. Ksentini and P. Frangoudis. 'Follow-Me Cloud: When Cloud Services Follow Mobile Users'. In: *IEEE Transactions on Cloud Computing* (2018), pages 1–1. ISSN: 2168-7161. DOI: 10.1109/TCC.2016.2525987.

[11] A. Panarello et al. 'Automating the Deployment of Multi-Cloud Applications in Federated Cloud Environments'. In: *Proceedings of the 10th EAI International Conference on Performance Evaluation Methodologies and Tools on 10th EAI International Conference on Performance Evaluation Methodologies and Tools*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2017, pages 194–201. ISBN: 978-1-63190-141-6. DOI: 10.4108/eai.25-10-2016.2266363.

[12] ETSI. *Network Functions Virtualisation (NFV); Management and Orchestration; Report on Architectural Options to Support Multiple Administrative Domains*. Group Report (GR) NFV-IFA 028 v3.1.1. European Telecommunications Standards Institute (ETSI), January 2018.

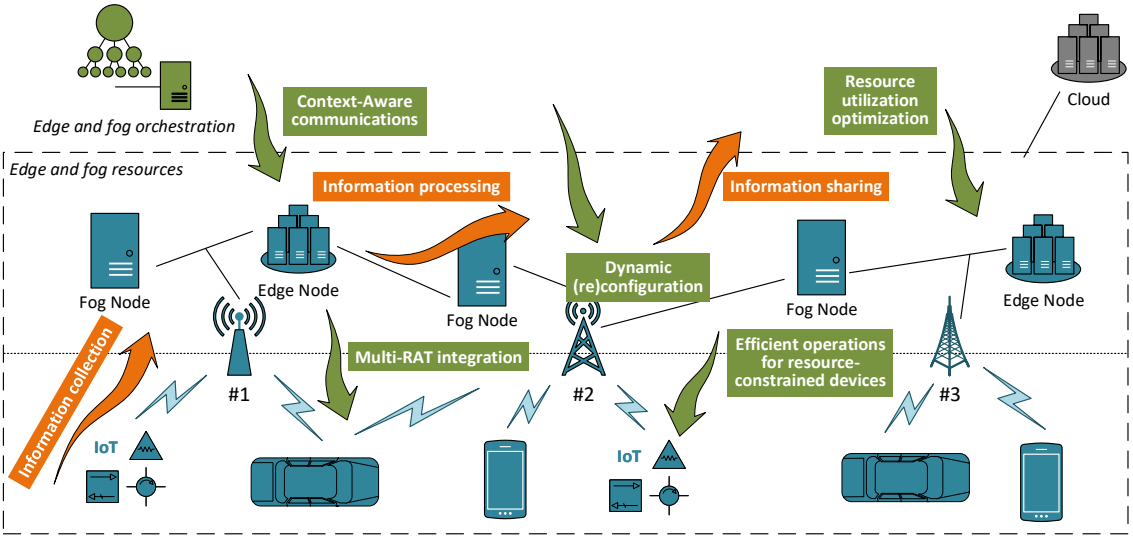


Figure 10.2: Edge and fog joint orchestration opportunities.

10.2 Opportunities of integrating edge and fog computing

By bringing fog computational resources into the vision of networking in 5G and beyond, several opportunities can be anticipated to enhance the system efficiency and performance. This section walks through a few potential benefits of joint edge and fog orchestration as illustrated in Figure 10.2.

10.2.1 Context-Aware Communications and Computations

Context-aware communications and computations open a new degree of freedom in optimising the network performance based on context information extracted from the underlying infrastructure of computing, storage and networking resources. This raises the opportunity of developing new algorithms to optimise the network performance based on the learning and intelligence derived from the context information of the edge and fog system. For example, where the edge and fog system is likely to have multiple co-existing RATs, one can envision efficient multi-RAT management and coordination algorithms by leveraging on the radio information extracted from each RAT. Artificial intelligence and machine learning based optimisation are examples of tools that can also be deployed in here.

10.2.2 Resource Utilisation Enhancement

Instead of solely relying on the computing substrates in the edge data centres, edge and fog orchestration allows the distribution of the various computing and networking tasks across both edge and fog resources, including any type of devices that possess networking and computing capabilities. This creates a larger pool of resources distributed near the end users enabling higher multiplexing gains, greater utilisation efficiency of the resources, and a larger pool of cooperating resources for executing certain functions or tasks tailored to the needs of the applications and end users. Such paradigm may create new business models wherein terminal devices can also participate in the pool of edge and fog resources in return of incentives (e.g., service subscription reduction), which helps infrastructure providers decrease the deployment and maintenance cost of their edge data centres.

10.2.3 Efficient Operation for Resource-Constrained Devices

In 5G, various categories of devices are envisioned. These range from vehicles and drones, to smartphones, tablets, and laptop computers, to IoT devices such as sensors or actuators. Clearly, some of these devices will have limited computational capability and battery (the so-called *resource-*

constrained devices) due to their low-cost nature. With an integrated edge and fog orchestration, these resource-constrained devices can now rely on the edge and fog resources to execute some of their computationally and power demanding tasks. This presents an opportunity for low-cost devices to remain intelligent and run advanced applications despite their limited capabilities.

10.2.4 Flexible and Scalable Functionalities

The edge and fog's NFV-inspired architecture and technologies aspire to become a viable proposition to enable flexibility and scalability of the envisioned 5G system. The foreseen joint orchestration would allow traffic engineering between nodes, thus setting roots to flexible behaviour and scalability of the system. Indeed, dynamic allocation of the computing and networking resources can be used for example to prioritise edge and fog resources in an area of higher demand which may lead to more optimised resource utilisation. A concentrated traffic or computing request can be directed to a limited number of edge and fog resources while others will shift to idle mode or be switched off, thus improving the overall energy conservation of the system. Furthermore, software migration and placement capability of the orchestration process allows for a seamless transfer of intelligence between geographically disparate nodes. This tackles the variable application delay constraints. Software components can be placed in the user vicinity fulfilling its latency requirements. Together with the ability to create interrelation (i.e., chain) of functions and applications and then map it into an underlying substrate of computing and networking resources, such ecosystem will be able to handle any 3rd party driven dependency between functions and applications whilst preserving the scalability of the solution at the same time.

10.3 Challenges of integrating edge and fog computing

The integration of edge and fog computing is still a new concept and it does not have yet a corresponding framework defined. Such framework will need to satisfy requirements of real-time communication utilising edge nodes, federation among multiple stakeholders, and dynamic resource discovery of volatile and non-volatile resources. Deployment strategies are also needed such as where to place the workload, connection policies, and when to use edge or fog nodes accounting for their heterogeneity. To define this framework, the following seven research challenges are identified.

10.3.1 Federation mechanisms

Federation is a process where different entities negotiate terms and conditions with a goal to form an alliance of trust and start sharing resources between each other. The result, the federation, should be beneficiary for all included entities. The key elements are the trust between entities and maintenance of negotiated conditions for long-term federation. Enabling trust between different entities is a challenge that can be solved using centralised or decentralised solution. The centralised solution is through single dedicated entity (server, repository) managed by the trusted organisation. It demands high level of maintenance and strict security policies. Additional resources may be needed to ensure scalability of the system. The decentralised solution is through a peer-to-peer network of trusted entities that maintain highly distributed repository. Although it considers a complex set-up operation and high-security risks, recent advances in trust-enabling technologies (e.g., Blockchain, Bitcoin, Ethereum) prove the contrary [MMM17][13], [Swa15][14]. The distributed repository can be deployed fast using current infrastructure as well as secure set-up of the peer-to-peer network of entities. The distributed repository is in line with the edge and fog architecture where the risks can

[13] E. Münsing, J. Mather and S. Moura. 'Blockchains for decentralized optimization of energy resources in microgrid networks'. In: *2017 IEEE Conference on Control Technology and Applications (CCTA)*. August 2017, pages 2164–2171. DOI: 10.1109/CCTA.2017.8062773.

[14] T. Swanson. *Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems*. White Paper. R3 CEV, April 2015.

be addressed through the scalability, storage and speed of the peer-to-peer solution (i.e., Blockchain) or through a combination of smart contracts, REST APIs and web applications (i.e., Ethereum).

Federation mechanisms are fundamental for the dynamic integration of multiple administrative domains into a unified platform using different granularity in either centralised or decentralised fashion. Different stakeholders expose different capabilities depending on their physical constraints (computing, storage, bandwidth) using different policies. It determines the stakeholders' degree of trust and conditions in which they are willing to join in the federation. The centralised solution holds the trust in a single entity. The terms and conditions are negotiated at a single point, exposing security threat (i.e., of a single point of failure) which by default demands maintenance and redundancy (e.g., similar to the Domain Name System (DNS) architecture). The decentralised solution distributes the trust burden to all entities. In this case, overlay peer-to-peer networks can be established between different stakeholders. A stakeholder can maintain several federation networks based on the degree of trust it exposes to each federation (e.g., gold federation, silver federation, etc.) as proposed in [Sim+16]. Using one or both approaches, the challenge is to rapidly and efficiently enable the edge and fog systems to dynamically scale up into unique virtualisation environment using the heterogeneous and exposed resources thus satisfying user and network demand in highly secure and trusted manner.

10.3.2 Dynamic discovery of resources

As aforementioned, the edge and fog computing system can be constructed by federating resources via joint orchestration. However, in contrast to edge or cloud computing where the tasks are basically performed by dedicated and static data centres or servers, fog computing could be carried out by mobile and battery-constrained devices that are volatile (i.e. may become available or unavailable spontaneously). Thus, it is key for the orchestration and management system to localise and monitor the available computing and networking resources, the pool of which may consist of both volatile and non-volatile substrates. In particular, the system should be able to identify the resources that have become available for federation, and such identification process is dubbed as *discovery* in this context. As the resources to be discovered may be physical devices that belong to different owners and/or administrative domains, there are two foreseeable challenges. First, how can devices discover or be discovered by the orchestration and management system?

Devices and associated resources may belong to different owners, have stationary or mobile nature, may have different availability or simply communicate by different protocols. The monitoring entity needs means to reach to those heterogeneous devices but also to estimate their stability and trust level in order to support stability of the overall system. Resource discovery in multi-RAT environments may require the use of distinct mechanisms depending on the connectivity availability. For instance, Link Layer Discovery Protocol (LLDP) works well in Ethernet networks while is not applicable to mobile networks (e.g., LTE). Similarly, IP-based mechanisms may not work in environments where Layer 2 security mechanisms are in place (e.g., 802.11 wireless networks). Second, how can resources be discovered across different (often overlapping) administrative domains? Different management systems of separate administrative domains need to have means to discover each other's resources in order to provide services which require enhanced pool of computing resources as well as to achieve better overall utilisation of overall pool of resources (e.g., reduce the energy usage during the idle periods in the network). To establish mutual resource usage, two control planes of two administrative domains must discover each other first. This process is a precondition for further federation process between different systems.

10.3.3 Multi-tenancy

Multi-tenancy refers to the support of co-existing applications requested by different tenants within the same infrastructure. All tenants perceive their resources as dedicated without mutual interference. Multi-tenancy enhances resource utilisation and enable business opportunities, which may find its application in edge or fog system alone. It is of greater need in federated edge and

fog systems for several reasons. First, there may be multiple over-the-top service (e.g., video, voice, social applications, etc.) providers who operate solely on top of the federated edge and fog systems. Second, there may be multiple industry vertical market players (electricity utility, automotive, e-health, etc.) who exploit federated edge and fog system to enhance system reliability and robustness. However, enabling multi-tenancy demands mechanisms to ensure security, isolation, and privacy among tenants. For network infrastructure, this is known as network slicing. For the edge and fog infrastructure, similar schemes are required to provide an isolated amount of cloud capacity customised to best suit specific application needs. More specifically, we need the following functions in federated fog and edge systems: (*i*) a function that performs admission control based on tenant's need (i.e., Service Level Agreement (SLA)) and current status of the infrastructure; (*ii*) a function that securely exposes selected service capabilities and management policies with a standard resource descriptor to the tenants for SLA negotiation and matching; and (*iii*) a function that provides performance monitoring information to the tenants.

10.3.4 Multi-virtualisation technology coexistence

Virtualisation refers to the different approaches for creating a virtual version of networking or computing hardware. There are multiple virtualisation techniques whose difference primarily resides in the location of the virtualisation layer and the way resources are used. Full-virtualisation provides a complete abstraction of the physical hardware. This allows software to run on distinct types of hardware without requiring any modification. This is the case of virtual machines. Hybrid-virtualisation provides an incomplete abstraction of the hardware. This imposes targeted modification to the software to run on different systems. This is the case of virtual machines using specific I/O hardware acceleration extensions. Para-virtualisation allows software to be executed in isolated domains but does not provide any hardware abstraction (e.g., software is explicitly written for a given operating system). This is the case of containers. A mix of those virtualisation techniques could be present at the same time in the edge and fog domains, especially if the overall ecosystem is the result of federation among multiple organisations. This poses a considerable challenge to the possibility of deploying any application on any node. Indeed, the orchestration system can instantiate applications only if the virtualisation substrate is compatible with the application packaging, thus reducing the possibilities of resource optimisation. Therefore, developers should package their applications for any possible target system. Such requirement could be relaxed by the usage of automated tools which take care of packaging the same application for multiple target systems. Tools like Vagrant[15] and the foreseen evolution could be hence extended to support multiple virtualisation substrates.

10.3.5 Functions and applications placement

Servers used to host applications/functions have a finite amount of compute capacity, notably in the case of fog, where resources are on the move and with limited compute capacity. In principle, services can be composed by placing functions and applications into the appropriate edge and fog Point of Presence (PoP). This requires provisioning of the computing resources. There will be cases where the functions and applications will be hosted in different domains (i.e., in the case of federation) and therefore it is necessary to decide which node to be used as edge or fog PoP. Typically, each request will have SLA requirements of latency, throughput and availability targets. The fundamental challenge is to deploy the functions and applications on integrated edge and fog resources while meeting the necessary SLAs. Several issues complicate the optimisation of functions and applications placement. First, volatility of the resource in terms of availability, for how long the resource will be available to be part of a service. Second, the workload varies dynamically and with the limited and finite compute capacity, it also becomes difficult to scale the resources up and down depending on the load. Third, there are performance issues that appear by co-locating the virtualisation functions and applications onto the same server or node. Fourth is

[15] Vagrant: `https://www.vagrantup.com/`

the complexity of optimal placement in a federated environment. Also in dynamic environments, resources may get fragmented which makes it more difficult to place optimally functions and applications. Integer Linear Programming (ILP) is often used for finding the optimal placement of functions and applications in static environments. However, those algorithms require a long process time before reaching to the solutions. To this end, heuristic algorithms are more suitable in dynamic environments where the solution, even if suboptimal, needs to be provided in a short time frame. Machine learning could be employed to enhance the accuracy of the placement based on historical data.

10.3.6 Dynamic service placement and migration

Service placement and migration is the process of transitioning an individual or organisational data across multiple cloud providers. With the advent of edge and fog computing, edge and fog nodes become places that users use to have seamless connection to the services with low communication latency. However, edge and fog computing brings in yet another challenge of dynamic service placement and migration. As a user moves to different geographical areas, should its service be migrated from one edge or fog node to another? The main challenge introduced here is to maintain relatively low service downtime and overall migration time without impacting the Quality of Service (QoS). It is challenging to find the optimal decision also because of the uncertainty of the user's mobility along with the transmission cost. In addition, the placement of the selected services needs to consider potential mobility patterns, to provide the desired performance to the associated user always.

10.3.7 Dynamic Resource Management

Dynamic resource management is the ability to manage dynamically the resources (i.e., compute, network, storage) by means of automation and self-allocation mechanisms. In a multi-RAT environment, one could always think about routing the traffic dynamically from one RAT to another depending on the user's/network's demand. In addition, probabilistic assumption on the mean workload needs to be derived at different time resolutions to provide the optimal compute/network resources to the users. One important challenge here is how to manage the fog and edge resources dynamically. This is especially challenging due the heterogeneity and volatility of the edge and fog resources.

10.3.8 Security

Any entity involved in the edge and fog computing can be possibly malicious, so security issues of the orchestration may mainly come from three aspects involving different entities' interactions: integrating heterogeneous platforms, sharing resources among devices, and hosting third-party applications. They require the authentication between different entities, dynamic resource authorisation, and the protection against malicious applications, respectively. In addition, those solutions designed to interwork with the cellular network require to be compliant with the 3rd Generation Partnership Project (3GPP) standard security requirements. It can be challenging to fulfil the requirements while keeping the edge and fog computing transparent to the 3GPP network architecture [Kek+18][16]. To prevent security threats of the edge and fog computing platforms from propagating towards the existing cellular network, the orchestration shall also provide a firewall-like security middleware between them. Though the software/hardware entities involved in the edge and fog computing solutions can be diverse, the orchestration shall introduce a set of general security requirements and mechanisms to establish a baseline security level.

[16] S. Kekki et al. *MEC in 5G networks*. White Paper 28. European Telecommunications Standards Institute (ETSI), June 2018.

11. An integrated edge and fog architecture

Based on the identified opportunities and challenges (see Chapter 10.2 and Chapter 10.3), a suitable approach for integrating edge and fog computing is based on a hierarchical infrastructure spanning across multiple tiers as illustrated in Figure 11.1. Such architecture comprises clouds and central data centres (Cloud/Central DCs) on top, edge data centres (Edge DCs) in the middle, and fog computing devices (Fog CDs) that are available locally in the access area. Central DCs are large scale public/operator-owned data centres while Edge DCs are small scale computing infrastructure deployed at the edge (e.g., fewer servers). Finally, Fog CDs comprise a variegated set of resources with limited computing capabilities like network nodes, end user devices, etc. The scope of the proposed architecture is on the edge/Fog tiers of the distributed computing infrastructure, along with their interaction with the distant tiers (e.g., cloud data centres).

The edge and fog tiers are therefore merged into a single computation platform, dubbed as Edge and Fog computing System (EFS), which serves as the environment for hosting virtualised functions, services, and applications. On the other hand, to manage, control, and federate resources for the EFS and its interaction with any other tiers, the architecture envisions an Orchestration and Control System (OCS) as another pillar component. These two logical entities, namely EFS and OCS, as well as their correspondent physical infrastructures (i.e., networking and computing substrates), are illustrated in Figure 11.1. The logical architecture of the EFS and OCS components is further detailed in Figure 11.2 which shows the various building blocks composing the overall architecture. The EFS and OCS references points are the results of an harmonisation of the several concepts proposed in European Telecommunications Standards Institute (ETSI) Network Function Virtualisation (NFV), ETSI Mobile Edge Computing (MEC), and OpenFog and are detailed later in Chapter 11.4. The following gives an overview of these different building blocks of the system and the architecture and explains how it offers enabling mechanisms of multi-Radio Access Technology (RAT) convergence. Finally, it identifies the similarities and differences between the proposed reference points and the ones defined in ETSI NFV and ETSI MEC.

11.1 The Edge and Fog computing System

The EFS is a logical system subsuming edge and fog resources in the edge and fog domains, and the networking substrate which may span from end user devices and things, to access nodes belonging to different RATs, further up to transport (i.e., fronthaul and backhaul) switches at the edge. The computing substrate of the EFS, on the other hand, is distributed across Fog CDs (e.g., stationary or mobile) and Edge DCs. The networking and computing resources composing the EFS may be owned by a plurality of different parties. These resources may not be directly interconnected, and may have different capabilities depending on their physical constraints (e.g., bandwidth, computing)

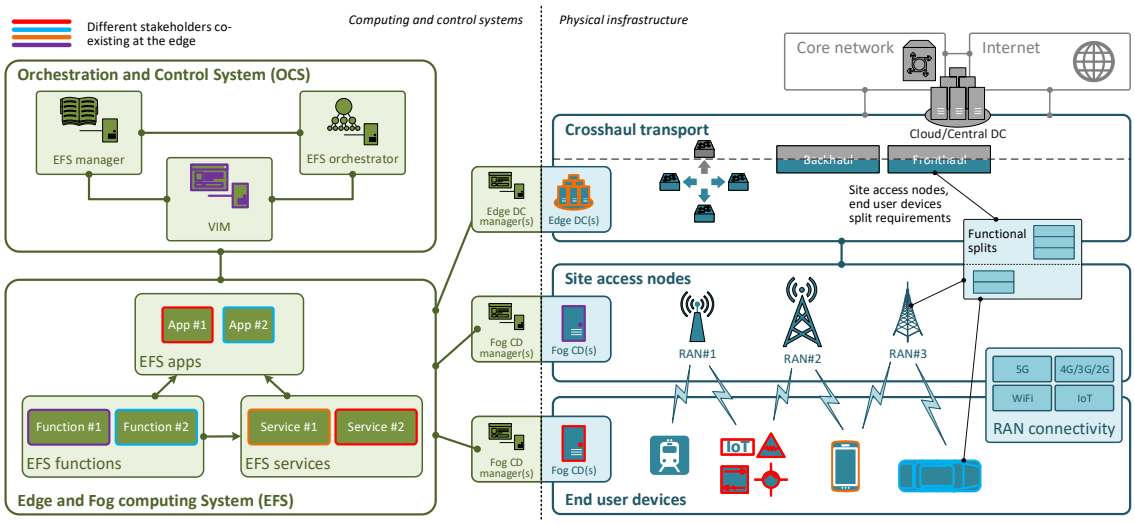


Figure 11.1: Integrated edge and fog concept including physical network and computing infrastructure.

or on the owners' policies (e.g., trust relationships). As shown in Figure 11.1, the EFS mainly hosts three types of components, namely *EFS services*, *EFS applications*, and *EFS functions*. This is akin to the ETSI MEC architecture and its components are described in the following.

11.1.1 EFS services

The *EFS services* are presented as context information that can be obtained from the EFS resources, but also from non-EFS domains such as the transport and core networks, as well as distant clouds and data centres. Context information from the EFS domain can be collected from both RAN elements and computing resources. Such information is abstracted and exposed through services running in the EFS for EFS applications (which will be discussed later) to consume. In order to enable the production and consumption of such information, an EFS service platform (see Figure 11.2) is envisioned with the goal of providing a publish/subscribe mechanism for service advertisement and registration via the reference point E2. For example, an enhanced localisation service could be offered out of context information consolidated across multiple RATs. Finally, the services available in distinct EFS may be federated together via the reference point F1 so as to increase the service footprint in multiple locations and administrative domains.

11.1.2 EFS applications

The *EFS applications* can subscribe to one or more EFS services through the E2 reference point. These applications can be either user or third-party applications. User applications refer to applications directly consumed by the user (e.g., Augmented Reality, User-Targeted Advertisements). A third-party application, on the other hand, is employed by certain vertical industries or products for various types of purposes. For example, a car safety application may be launched to give pre-crash warnings and collision avoidance signalling in a timely manner, by fetching precise information relating to the location of nearby vehicles with sufficiently low latency. Another example for third-party application is an Internet of Things (IoT) gateway application to facilitate the coordination and management of IoT devices and enhance the overall security, as well as using context information collected from IoT sensors to optimise the performance of other EFS functions and applications.

11.1.3 EFS functions

The *EFS functions* are mainly networking functions (e.g., for the access, transport, and core domains) that are beneficial to run in the EFS to optimise the network connectivity service Key

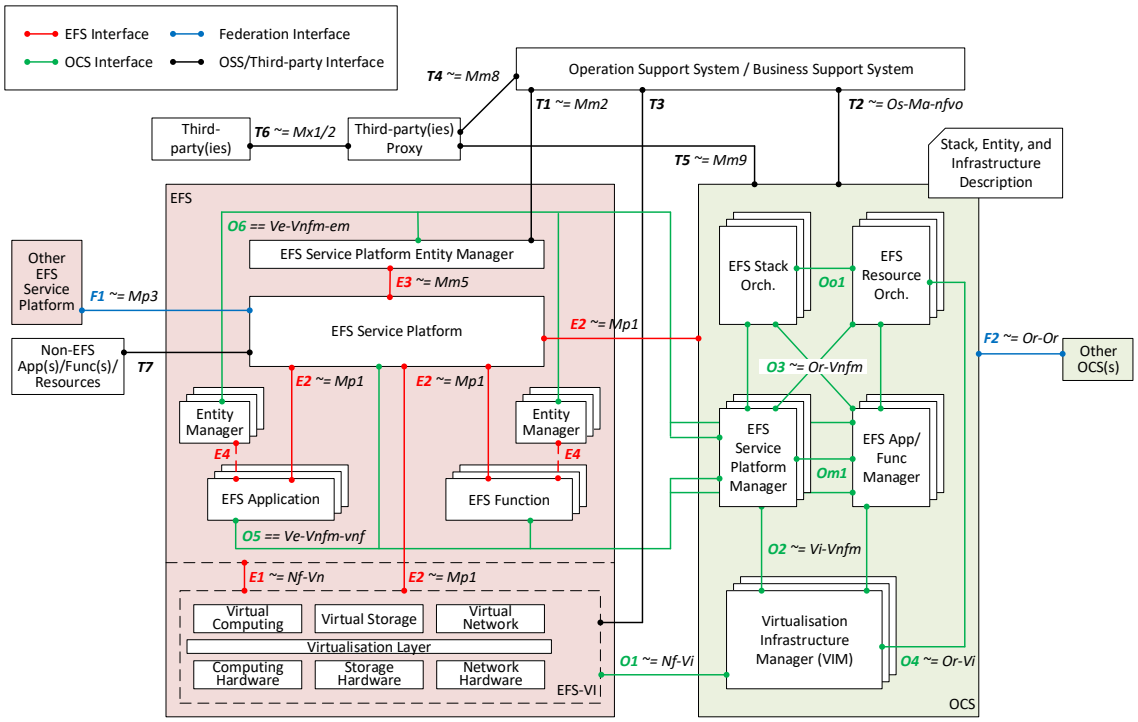


Figure 11.2: Integrated edge and fog computing architecture detailing EFS and OCS components.

Performance Indicator (KPI) and expose target context information via the E2 reference point. In the Radio Access Network (RAN), including 3rd Generation Partnership Project (3GPP) and non-3GPP RANs, Virtualised RAN (V-RAN) functions (e.g., an access node implementing a protocol functional split) could find it beneficial in terms of latency to execute in the EFS instead of a distant data centre. However, not all V-RAN functions are suitable to run in the EFS as there are clearly trade-offs to be made: latency and bandwidth gains at the edge *vs* pooling, multiplexing, and coordination gains at central locations. Given the EFS pervasiveness, the principle of RAN functional split may now be applied to the end user devices and things when densely interconnected, and no longer an exclusivity to the access nodes as it is today. This also suggests an evolved form of Device-to-Device (D2D) networking for discovery and establishment of direct connections to nearby devices or nodes (e.g., Fog CDs) that offer virtualisation capabilities and context information services.

A finer granularity of the RAN functional split could be also envisaged, where instead of splitting between stack layers or functions in the same layer as today, the splitting could be done inside the same function into elementary virtualised functions executing in tailored EFS resources. For example, this could be the case of authentication functions whose decomposition and virtualisation could be advantageous to facilitate seamless session continuity between different RATs. In addition to the RAN, Virtual Network Function (VNF) from the transport and core could also find hosting in the EFS so as to give the local access all what is needed to keep its traffic locally. For example, some core network functions (e.g., Mobility Management Entity (MME) and Packet Data Network Gateway (PGW) in 4G networks) virtualised closer to the users can facilitate data offloading and mitigate heavy-signalling caused by frequent handovers in the local access.

11.2 The Orchestration and Control System

The OCS is a logical system in charge of composing, controlling, managing, orchestrating, and federating one or more Edge and Fog computing Systems. Therefore, it has the mission to deliver a flexible management and control capable of coping with the dynamicity and heterogeneity of the EFS (e.g., as user conditions change, network conditions vary, and part of the involved resources

might become unavailable or even move). Therefore, it shall guarantee scalability and seamless interoperability with other domains (e.g., transport, core, distant clouds, etc.). Moreover, an OCS may interact with other OCS domains. As shown in Figure 11.1, three main logical entities compose the OCS, namely (*i*) the Virtualised Infrastructure Manager (VIM), (*ii*) the EFS manager, and (*iii*) the EFS orchestrator.

11.2.1 Virtualisation Infrastructure Manager

The *VIM* comprises the functionalities that are used to control and manage the interaction of the EFS service platforms, EFS functions, and EFS applications with the edge and fog resources under its authority, as well as their virtualisation. Multiple VIMs may be deployed to control and manage distinct virtualisation substrates or administrative domains so as to offer a unified virtualised execution environment view. Distinct execution environment are expected to coexist, such as hypervisors for virtual machines, containers, as well as native applications on resource-constrained devices. The OCS also takes care of controlling the inter-connectivity of the EFS resources, which might not be co-located within the same connectivity domain and could be connected by different types of networks. The VIM exposes and makes use of the O1, O2, and O4 reference points as well as the E2 reference point.

11.2.2 EFS manager

The *EFS manager* is responsible for the life cycle management (e.g., instantiation, update, query, scaling and termination) of the service platforms, functions, and applications in the EFS. Multiple EFS Managers may be deployed to manage distinct components of the EFS (e.g., service platforms, functions, and applications) whereas each EFS manager may manage a single service platform, function, application or a pool of them. Given the inter-dependency of applications/functions and the publish-subscribe model adopted by the service platform, two EFS managers are defined at architectural level: one dedicated to the management of the service platform and one dedicated to the management of the applications and functions. The EFS managers expose and make use of the O2, O3, O5, O6 and, Om1 reference points, as well as the E2 reference point.

11.2.3 EFS orchestrator

The *EFS orchestrator* is in charge of the orchestration and management of edge and fog resources and composing the EFS. An EFS orchestrator comprises an EFS resource orchestrator and an EFS stack orchestrator as detailed in Figure 11.2. An *EFS resource orchestrator* supports accessing the edge and fog resources in an abstracted manner independently of any VIM, as well as governance of EFS service platform, EFS function, and EFS application instances sharing resources in the EFS. An *EFS stack orchestrator* is responsible for the *EFS stack* life cycle management operations such as on-boarding, instantiating, and terminating a stack. The EFS orchestrator expose and make use of the O3, O4, Oo1, T2, F2, and T5 reference points, as well as the E2 reference point. Specifically, the reference point F2 (see Figure 11.2) can be used to dynamically federate resource thus allowing the EFS to scale so as to satisfy the network and user demands.

An *EFS stack* can be viewed architecturally as a forwarding graph of functions and/or application interconnected by supporting edge and fog resources and/or EFS service platforms. An EFS stack extends the ETSI NFV *network services* by also considering interconnections with applications and service platforms and not only between network functions. In order to enable that, the *EFS stack descriptor* extends the *ETSI NFV network service descriptor* by also considering applications and service platforms in addition to network functions. It describes the requirements and interconnections of one or more EFS functions and EFS applications between them or with the EFS service platform. Finally, the *EFS entity descriptor* extends and combines the *ETSI NFV VNF* and *ETSI MEC App* descriptors so as to uniformly describe the various characteristics of EFS functions, EFS applications, and EFS service platform. EFS entity descriptors are referenced and included into an EFS stack descriptor so as to allow the EFS orchestrator to properly deploy all the

entities and interconnect them.

11.3 Application to multi-access convergence

As it can be evinced from the above, a flexible EFS governed by OCS is positioned as a perfect platform for multi-RAT convergence. To be specific, EFS functions may embrace *network optimisation algorithms* that can subscribe to EFS services (e.g., context information from multiple RATs) to optimise the network performance in accordance with the specific context of the given local access area. For example, a multi-RAT convergence function could be envisioned in the EFS to optimise the traffic delivery across multiple RATs based on the context information of end users and access nodes (e.g., location, load, mobility, etc.). Offloading from one RAT to another, adapting a RAT configuration based on context information from another RAT, aggregation across RATs, assistance between RATs, sharing between RATs (e.g., License Shared Access), tight coordination between nodes and devices from multiple RATs, are all examples of options that can be considered in such a function. For example, consider licensed-assisted access (LAA) mechanisms in Long Term Evolution (LTE), a network node (i.e., base station and UE) needs to follow a listen before talk (LBT) procedure to ensure the unlicensed band is clear before accessing such resource. By allowing both licensed-band (LTE) and unlicensed-band (Wi-Fi) RATs to share the information with each other in the same platform, it may potentially reduce the interference level and improve the utilisation of the unlicensed bands by certain coordination and optimisation.

11.4 Reference points: similarities, differences, and extensions

This section provides an overview of the reference points of the integrated edge and fog architecture illustrated in Figure 11.2. Specifically, the following identifies the relationship of the proposed reference points with ETSI NFV [ETS13b][1] and ETSI MEC [ETS16a] specifications, along with their functionalities, similarities, differences, and the envisioned extensions. Please note that an exemplification of some of the reference point is provided in the following Chapter 11.5.

E1 — ETSI NFV: Nf-Vn

This reference point represents the execution environment provided by the EFS virtualisation infrastructure, and supports the following:

- Virtual machine-based hypervisors;
- Container-based hypervisors;
- Native application environments (e.g., Linux, Windows, etc.) .

Compared to the ETSI NFV environment, such reference point also supports native applications for resource-constrained devices.

E2 — ETSI MEC: Mp1

This reference point is used for exchange between the OCS and the EFS service platform and supports the following:

- Forwarding of configuration information, failure events, measurement results, and usage records regarding edge and fog resources for monitoring purposes.

Compared to the ETSI NFV environment, such reference point is added to support the publication and consumption of data from and to the EFS service platform. This is similar to what is done in ETSI MEC for service registration, service discovery, and communication support for services. Nevertheless, this reference point also supports services to be published and/or consumed by physical/virtual resources, functions, and OCS components in addition to the sole ETSI MEC applications.

[1] ETSI. *Network Functions Virtualisation (NFV); Architectural Framework*. Group Specification (GS) NFV 002 v1.1.1. European Telecommunications Standards Institute (ETSI), October 2013.

E3 — ETSI MEC: Mm5

This reference point is used for configuring the EFS service platform configuration, and supports the following:

- Configuration of the EFS applications/functions rules, including authorisation and requirements, for EFS service publication and consumption.

Compared to the ETSI MEC environment, such reference points does not support EFS application/function relocation and life cycle procedures which are instead delegated to the OCS.

E4 — Not specified

This reference point is an internal reference point between the EFS applications/functions and their corresponding entity managers. While this reference point is relevant at architectural level, it is usually a proprietary reference points since it is tightly coupled to EFS applications/functions implementation and functionalities. E3 is a realisation of the E4 reference point where a set of functionalities has been defined for the EFS service platform.

This reference points is also reported but not specified in the ETSI NFV environment.

F1 — ETSI MEC: Mp3

This reference point is used for exchange between different EFS service platforms when federation at EFS level is in place. Such reference point supports the functionalities defined for E2 and E3 reference points. Federation agreements may limit the functionalities over this reference point, either in availability or scope.

Compared to the ETSI MEC environment, such reference point also supports the exchange of EFS services in addition to the control communication between ETSI MEC platforms.

F2 — ETSI NFV: Or-Or, Vi-Vi

This reference point is used for exchange between different OCSs when federation at OCS level is in place. Such reference point supports the functionalities defined for O2, O3, O4, Om1, and Oo1 reference points. Federation agreements may limit the functionalities over this reference point, either in availability or scope.

Compared to the ETSI NFV environment, such reference point also supports the interaction between EFS managers in the scope of EFS service platform.

O1 — ETSI NFV: Nf-Vi

This reference point is used for exchanges between the VIM and the edge and fog resources composing the EFS and supports the following:

- Allocation of functions and applications with indication of compute/storage resource;
- Update, migration, and termination of functions and applications including their resource allocation;
- Creation, configuration, and termination of connection between functions, applications, and service platform.

Compared to the ETSI NFV environment, such reference point also needs to support intermittent connectivity on volatile low-end devices with heterogeneous virtualisation support.

O2 — ETSI NFV: Vi-Vnfm

This reference point is used for exchanges between the VIM and the EFS managers and supports the following:

- Edge and fog resources information retrieval;
- Edge and fog resource allocation and release;
- Notification from the VIM to the EFS manager of events, measurement results, and usage records regarding edge and fog resources used by a specific application or function.

Compared to the ETSI NFV environment, such reference point also needs to support information regarding mobility and battery-level of resources in addition to information regarding privacy constraints and negotiated Service Level Agreement (SLA)s.

O3 — ETSI NFV: Or-Vnfm

This reference point is used for exchange between the EFS orchestrator and the EFS manager, and supports the following:

- Edge and fog resources authorisation, validation, reservation, and release for a function or application;
- Edge and fog resources allocation/release request for an application or function;
- Application and function instantiation, update, termination, scaling in/out and up/down;
- Application and function instance query to retrieve any run-time information.

Compared to ETSI NFV environment, such reference point also needs to support information regarding federation and privacy constraints of underlying resources.

O4 — ETSI NFV: Or-Vi

This reference point is used for exchange between the VIM and the EFS orchestrators and supports the following:

- Edge and fog resources information retrieval;
- Edge and fog resource allocation and release;
- Function and application addition, deletion, update;
- Notification from the VIM and the EFS orchestrator of events, measurement results, and usage records regarding edge and fog resources.

Compared to the ETSI NFV environment, such reference point also needs to support information regarding mobility and battery-level of resources in addition to information regarding privacy constraints and negotiated SLAs.

O5 — ETSI NFV: Ve-Vnfm-Vnfm

This reference point is used for exchange between functions or applications and the corresponding EFS application/function manager, or between the EFS service platform and the corresponding EFS service platform manager, and supports the following:

- Notification from the application/function/service platform to the corresponding EFS manager of events, measurements, and usage records regarding the application/function/service platform itself;
- Verification that the application or function is still alive/functional.

This reference points does not requires any change compared to the ETSI NFV environment.

O6 — ETSI NFV: Ve-Vnfm-em

This reference point is used for exchange between the entity managers of functions or applications and the corresponding EFS application/function manager, or between the EFS service platform entity manager and the corresponding EFS service platform manager, and supports the following:

- Application/function/service platform instantiation, update, termination, scaling in/out and up/down;
- Application/function/service platform instance query to retrieve any run-time information;
- Configuration of events and measurements regarding the application/function/service platform itself;

This reference points does not requires any change compared to the ETSI NFV environment.

Om1 — New

This reference point is used for exchange between EFS application/function manager and EFS service platform manager, and supports the following:

- Notification of functions and applications information regarding instantiation, update, termination, migration, and scaling in/out and up/down with regards to their publication and subscription of services on the EFS service platform;
- Notification of information regarding the availability of services at given locations.

Compared to the ETSI NFV environment, such reference point is added to support the interdependency between functions, applications, and the service platform. The life cycle management

of functions and applications requires coordination with the EFS service platform so as to satisfy the agreed SLAs.

Oo1 — New

This reference point is used for exchange between the EFS resource orchestrator and the EFS stack orchestrator, and supports the following:

- Notification of events, measurement results, and usage records regarding edge/fog resources and EFS stacks.

Compared to the ETSI NFV environment, such reference point is added to support the dynamic environment where the mobility and volatility of resources are parameters that need to be considered for the life cycle management of EFS stacks.

T1 — ETSI MEC: Mm2

This reference point is used for exchange between the OSS/BSS and the EFS service platform entity manager, and supports the following:

- Configuration of the EFS service platform regarding OSS/BSS relevant aspects (e.g., charging for consumption of EFS services);
- Notification of events, measurement results, and usage records regarding EFS services published and/or consumed by EFS applications/functions.

Compared to the ETSI MEC environment, such reference point does not support the fault and performance management of the EFS service platform which are instead delegated to the OCS.

T2 — ETSI NFV: Os-Ma-nfvo

This reference point is used for exchange between the OSS/BSS and the EFS orchestrator, and supports the following:

- EFS stack descriptor and EFS stack life cycle management, including EFS stack instantiation, update, scaling, migration, termination, and query (e.g., retrieving summarised information about edge and fog resources associated to the EFS stack instance);
- Policy management and or enforcement for EFS stack instances, function and application instances, and edge and fog resources (e.g., authorisation, access control, resource reservation, placement, allocation, etc.);
- Forwarding of events, accounting and usage records and performance measurement results regarding EFS stack instances, application and function instances, and edge/fog resources to OSS/BSS, as well as and information about the associations between those instances and edge/fog resources;
- Integrating and releasing of resources into/from the target EFS including third-party information and SLAs.

Compared to the ETSI NFV environment, such reference point also needs to support federation and privacy of multiple administrative domains coexisting in the same environment.

T3 — Not specified

This reference point is an reference point between the EFS virtualisation infrastructure and its corresponding OSS/BSS. While this reference point is relevant at architectural level, it is usually a proprietary reference points since it is tightly coupled to the operational and business decisions of the stakeholder operating the EFS virtualisation infrastructure.

This reference points is also reported but not specified in the ETSI NFV environment.

T4 — ETSI MEC: Mm8

This reference point is used for exchange between the Third-party(ies) Proxy and the OSS/BSS and supports the following:

- Requesting the deployment of third-party applications/functions on the target EFS;
- Integrating and releasing of resources into/from the target EFS including third-party information and SLAs.

Compared to the ETSI NFV environment, such reference point is added to support interaction with third parties (instead of OSS/BSS) similar as done in the ETSI MEC for the MEC application deployment.

T5 — ETSI MEC: Mm9

This reference point is used for exchange between the Third-party(ies) Proxy and the EFS orchestrator and supports the same operations as T2. The difference resides in the two different endpoints: OSS/BSS in case of T2, Third-party(ies) Proxy in case of T5.

Compared to the ETSI NFV environment, such reference point is added to support interaction with third parties (instead of OSS/BSS) similar as done in ETSI MEC for the life cycle management of third-party applications.

T6 — ETSI MEC: Mx1/Mx2

This reference point is used for exchange between the OSS/BBS and the customer-facing service portal, and supports the following:

- Requesting the deployment of third-party applications/functions on the target EFS;
- Requesting the integrating and release of resources into/from the target EFS including third-party information and SLAs.

Compared to the ETSI NFV environment, such reference point is added to support the deployment of third-party functions similar as done in the ETSI MEC for third-party application. Moreover, this reference points also supports the integration and release of third-party resources which was not considered in ETSI MEC.

T7 — New

This reference point is used for exchange between the EFS service platforms and applications/functions/resources not belonging to the EFS domain. Such reference point supports the functionalities defined for the E2 reference point. A limited level of trust may be applied to the non-EFS applications/functions/resources, thus limiting the functionalities over this reference point, either in availability or scope.

Compared to the ETSI MEC environment, such reference point is added to support different level os trustiness of the applications/functions/resources.

11.5 An exemplary use case: from cloud robotics to fog-assisted robotics

This section first presents an exemplary use case benefiting from an edge and fog integrated system, namely fog-assisted robotics. Next, it applies the architecture illustrated in Figure 11.2 to the use case by exemplifying the environment, including the physical infrastructure, and the EFS functions, EFS applications, and EFS services. Finally, it presents an example of the interaction between EFS and OCS, including the use of context information, tailored to the dynamic migration of EFS applications.

11.5.1 Limitations of cloud robotics

Cloud robotics is a field of robotics that leverages and integrates cloud computing, cloud storage, and other Internet technologies, into industrial and commercial applications. Cloud technologies enable robot systems to be endowed with powerful capability by leveraging the powerful computation, storage, and communication resources available in the cloud. Consequently, it is possible to build lightweight, low cost, and smarter robots by placing an intelligent brain in the cloud which offers a converged infrastructure that can be also used to share services and information from various robots or agents. To that end, a cloud for robots shall support [Keh+15][2] the sharing of object data between various robots and agents connected to the cloud, such as images, maps, robot outcomes, trajectories,

[2] B. Kehoe et al. 'A Survey of Research on Cloud Robotics and Automation'. In: *IEEE Transactions on Automation Science and Engineering* 12.2 (April 2015), pages 398–409. ISSN: 1545-5955. DOI: 10.1109/TASE.2014.2376492.

and control policies. Moreover, on-demand provisioning of parallel computing resources is needed for motion planning, task planning, multi-robot collaboration, scheduling and coordination of the robotic system. Finally, on-demand human guidance and assistance, via also augmented human-robot interaction, is required for evaluation and error recovery.

Though robots can benefit from various advantages of cloud computing, this presents several limitations when applied to the cloud robotics field. cloud facilities traditionally reside far away from the robots and while the cloud providers can enforce SLA in their infrastructure, very little can be ensured in the network between the robots and the cloud. As a result, cloud-based applications can suffer from high-latency or unpredictable jitter in the network. For instance, controlling a robot's motion which relies heavily on sensors and feedback of controller is extremely challenging without assured network performance, especially when the traffic traverses many Internet Service Provider (ISP). Indeed, a fault in the network could leave the robot brainless and out of control. For that reason, tasks involving real-time execution require nowadays either on-board processing or a dedicated infrastructure close to the robots. The former solution is usually adopted when few robots are deployed in a given area and the cost of installing a dedicated cloud-like infrastructure on-site is prohibitive compared to the on-board processing. In this case, robot capabilities are typically more limited compared to a cloud-based solution. The latter solution instead is usually adopted when many cooperative robots are deployed in the same area and the benefits of cloud computing in terms of coordination overcome the costs of deploying a dedicated computing and networking infrastructure. This is the case of automatised warehouses where hundreds of mobile platforms are employed to move pallets. Notwithstanding, the two solutions are a palliative for today's cloud robotics and none of them can provide all the cloud computing benefits, including the usage of a converged infrastructure for sharing services and information. As result, a paradigm shift from cloud robotics towards fog-assisted robotics is needed.

11.5.2 The need of shifting towards the edge and fog for robotics systems

Computing and networking resources sprout in any location reaching a pervasive presence in today's environments. Devices like computers, laptops, APs, routers, base stations, smartphone, etc. are all around us, however their usage is limited (and restricted) to the sole and unique purpose they have been built for. This leads to a huge amount of independent and not integrated resources. Robots operating in a certain area could potentially make use of those resources to accomplish distinct tasks, especially the ones with stringent latency requirements, and take advantage of the services and information available locally. To exemplify such concept, let's consider a Shopping Mall environment which also serves as fog-assisted robotics reference scenario. A Shopping Mall traditionally comprises a variegate set of computing and networking resources, spanning from wireless and wired infrastructure (e.g., 802.11 APs, femto-cell, Ethernet backbone, etc.) to sensors (e.g., fire alarm, temperature, security cameras, etc.) and computing facilities (e.g., server room). Such heterogeneity presents a great chance for enhancing robot capabilities without the need of deploying an ad-hoc infrastructure. By hosting the brain close to the robot, proper performance can be ensured on the network and local context information as well as multiple connectivity options available on-site can be leveraged to accomplish complex tasks. However, to keep the benefits provided by the consolidated infrastructure at cloud level, fog-assisted robotics also require a converged platform in the edge and fog tiers. For the fog-assisted robotics use case, two exemplary scenarios are envisioned in the Shopping Mall:

1. In the first scenario, the robots are in charge of keeping clean the floors in common areas of the Shopping Mall, thus providing a cleaning service;
2. In the second scenario, the robots provide synchronised delivery of goods within the Shopping Mall building to restock the supplies of the several shops.

These scenarios require the real-time feeding of the robots with multiple inputs and data about the environment. For instance, to detect the dirty areas to clean as well as the various spills that regularly occur within the Shopping Mall, the robotic application needs to process the video streams

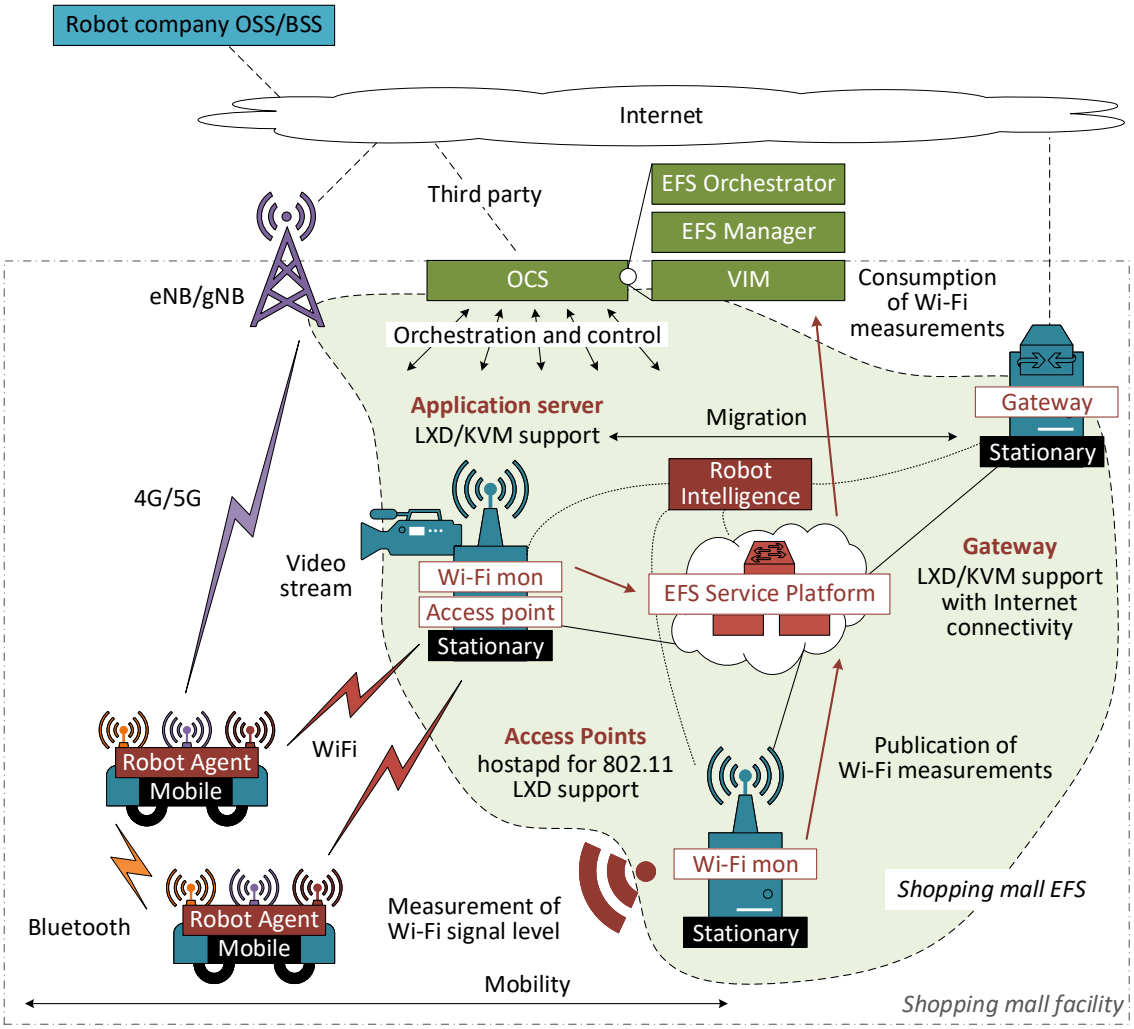


Figure 11.3: Exemplary fog-assisted robotics logical system in a Shopping Mall environment.

from multiple cameras distributed across the Shopping Mall building. Since multiple cameras are already available in the Shopping Mall for security reasons, there is no need to deploy ad-hoc cameras for the robot which in turn may process the raw video data available in the infrastructure at the edge and fog. Raw video is hence collected at the edge and fog computing platform and made available to the robotics application which can further process it via video analytics techniques to identify the areas to clean in a timely manner. Once a dirty area/spill has been positively identified, the robotic application necessitates an indoor navigation system to guide the cleaning robot to the precise location of the point of interest. In addition, the cleaning application may also leverage context information data available locally to estimate the number of people present along the path to be followed by the robot. This allows the brain to decide whether performing the cleaning operation can be risky (or not convenient) if some areas are particularly crowded and may hamper robot's movements. Finally, the brain residing in the EFS guides and instructs the robot to execute the cleaning task. The second scenario builds on top of the first one and contemplates multiple cooperating robots for resupplying the different shops. Data related to the stock level of each shop is collected and analysed at the EFS and is used to determine which good needs to be delivered to which shop. Some items may be too large for one robot alone and would require the synchronised operation of two or more robots to carry it. Thanks to the vicinity of the brain to the robots, it is hence possible to achieve tight coordination between the robots. Remarkably, the same navigation system and localisation service can be used to guide the robots to the shops without the need of deploying them twice.

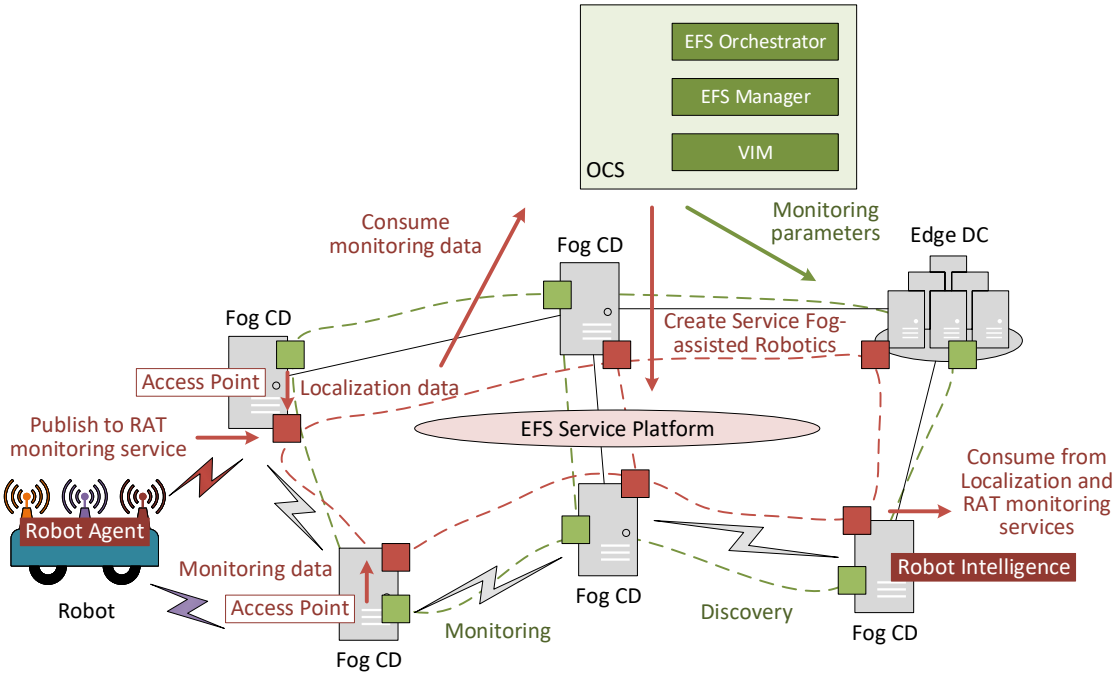


Figure 11.4: Potential physical mapping of integrated edge and fog system for the exemplary fog-assisted robotics use case.

11.5.3 Exemplifying the fog-assisted robotics edge and fog system

As it can be evinced from the description in Chapter 11.5.1 and 11.5.2, multiple computing and networking technologies are involved in these scenarios, as depicted in Figure 11.3:

- Cameras across the Shopping Mall could be either connected to the EFS via Ethernet or Wi-Fi. These cameras are usually special-purpose devices with enough capability to record and stream data to the EFS;
- The localisation service could be based on presence sensors which communicate over ZigBee, Bluetooth, or also via Wi-Fi. Data is sent periodically to the EFS which performs a local analysis;
- Robots connect to the EFS via Wi-Fi and/or LTE. In addition, robots may have Bluetooth chipsets for local connectivity;
- Video processing requires powerful computing platforms with video accelerators (i.e., GPU) to perform the necessary analytics;
- Localisation and navigation services instead require parallel computing depending on the amount of data (i.e., x86 servers);
- Robots and access points may offer limited computing capabilities (e.g., ARM boards) that can be used e.g. to instantiate networking functions (e.g., Wi-Fi APs, D2D).

The potential physical mapping of the integrated edge and fog system is illustrated in Figure 11.4, which highlights the distributed nature of both the EFS and the OCS.

All the physical resources need to be integrated into the same EFS platform. For the cleaning task, each component (cameras, sensors, robots) communicates via different RATs which are blended together for accomplishing a more complex task. A clear example of such multi-RAT cooperation is the localisation service which can leverage multiple connectivity technologies to determine the position of the robots. This is of particular relevance given the well-known shortcomings of Global Positioning System (GPS) when applied in indoor environments and the impossibility for the robots to rely on GPS signal for the navigation in the Shopping Mall. Furthermore, the robot requires multiple RATs simultaneously active to achieve the desired synchronisation level for jointly delivering large items. For instance, Bluetooth connectivity can be used for the feedback control loop between the different spatially-close robots. Low latency and jitter are critical requirements

to keep the robots aligned and synchronised when moving. Therefore, a direct communication (no hops) between the robots is desirable. A network-assisted D2D mechanism is hence required to be able to perform the Bluetooth pairing between the different robots, especially considering that different robot formations may occur at various times (e.g., a formation of two, three, or more robots require different pairing combinations). To achieve that, the Bluetooth pairing may use the primary Wi-Fi/LTE channel to initiate and configure the D2D communication.

The following reports the various EFS applications, EFS functions, and EFS services envisioned in the fog-assisted robotics use case.

Robot intelligence – EFS application

This EFS application is placed within the EFS and remotely controls the robots by implementing all the robotics intelligence. This includes:

- Receiving data from the sensors on the robots (e.g., motor encoders, bumpers, lidar, etc.) to implement a close-loop control mechanism;
- Computing the best route for the robots to reach a given area based on the task (e.g., cleaning service *vs* synchronised delivery);
- Consuming the localisation and Wi-Fi monitoring services to enhance the robotics operations (e.g., adapting the speed of the robots depending on the signal level);
- Communicating to the robots the set of instructions to execute;

This application does not require any access to the hardware, therefore it can be provided in form of Virtual Machine (VM) or container.

Robot agent – EFS application

This EFS application is placed on the robots, which are part of the EFS. The tasks of this application include:

- Reading the data from the sensors equipped on the robots (e.g., motor encoders, bumpers, lidar, etc.);
- Communicating the sensor data to the robot intelligence application;
- Receiving and executing the instruction computed by the robot intelligence application.

Given the requirement of this application to directly access the hardware (i.e., sensors and motors), this application is deployed in the form of container or native application. Please note that container platforms, if properly configured, can provide direct access to a subset of hardware devices.

Virtual Wi-Fi access point – EFS function

This EFS function implements a 802.11 wireless access point so as to enable the infrastructure-to-robot communication, which is essential for robot navigation. Commands to control the robot are sent over Wi-Fi connections managed by virtual APs, which allows seamless Wi-Fi connectivity for a roaming Wi-Fi client and avoid connection disruptions. This function is also employed to help robots establish Bluetooth D2D communication in case accurate movement synchronisation is required. Given the requirement of this application to directly access the hardware (i.e., Wi-Fi card), this function is deployed in form of container or native application.

Gateway – EFS function

This EFS function implements IP gateway capabilities so as to enable the robots to access the Internet and eventually communicate with the robotics company Operation Support System (OSS)/Business Support System (BSS). This also includes:

- Network Address Translation (NAT), Virtual Private Network (VPN), and firewall capabilities;
- Domain Name System (DNS) and Dynamic Host Configuration Protocol (DHCP) support for the robots applications and functions.

This function does not require any access to the hardware, therefore it can be provided in form of VM or container.

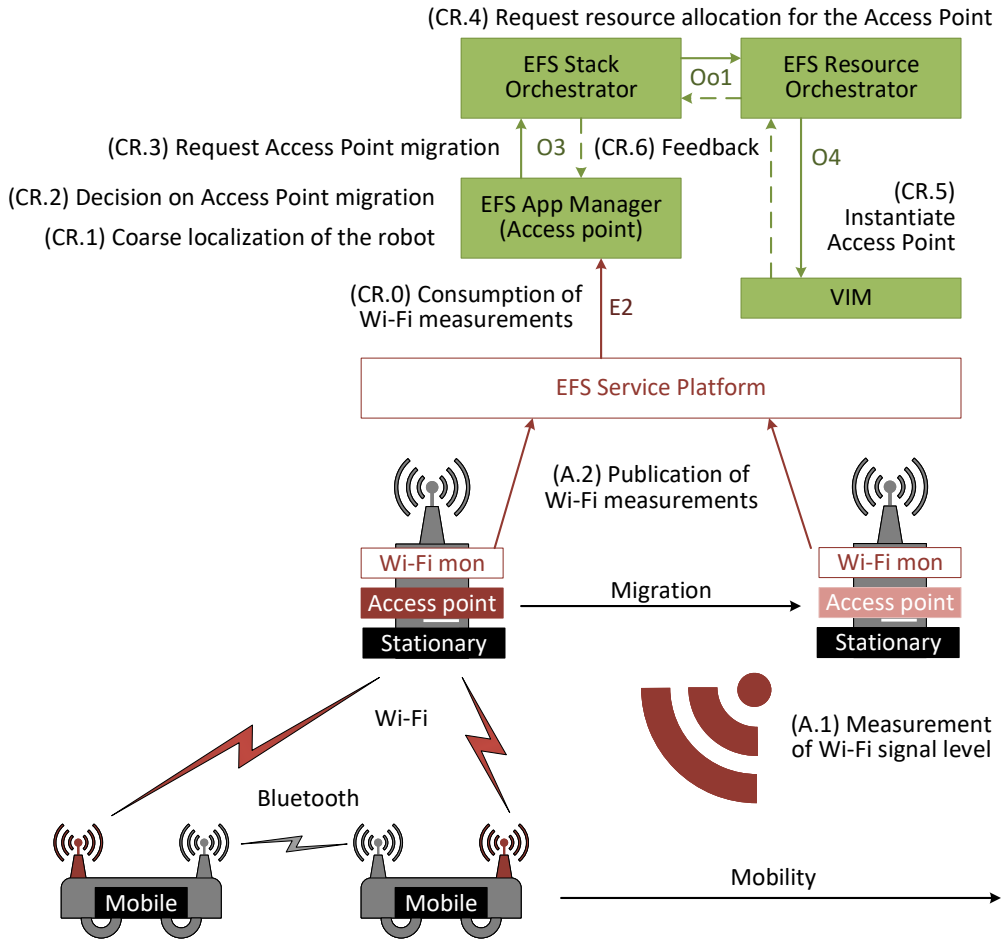


Figure 11.5: Interaction between EFS and OCS for a follow-me migration.

Wi-Fi monitoring – EFS function and EFS service

This EFS function implements Wi-Fi monitoring capabilities via the EFS service platform. This function runs on the resources equipped with Wi-Fi card and monitors the following parameters of Wi-Fi surrounding traffic and Wi-Fi stations:

- Station Wi-Fi channel;
- Station wireless signal level;
- Transmission and reception data rates at data link level;
- Number of retransmission and packet losses at data link level;
- Number of successfully transmitted/received bytes and packets.

The above information is hence published as EFS service and consumed by the robot intelligence application for determining (*i*) the status os the wireless link, and (*ii*) the coarse localisation of the robots with respect to the access points.

11.5.4 Interaction between EFS and OCS implementing a follow-me migration

In the considered fog-assisted use case, robots are connected via Wi-Fi and move in the Shopping Mall to accomplish different tasks (i.e., cleaning service and synchronised delivery). To that end, the robots require constant Wi-Fi coverage wherever they go. In the following example, the interaction between EFS and OCS is aimed at implementing a follow-me migration functionality, that is the virtual access point EFS function is migrated following the robots movement. To this purpose, an EFS function manager is deployed and dedicated to the virtual access point in order to detect the movement of the robots and trigger the migration of the EFS function so as to provide full connectivity coverage in the Shopping Mall. Figure 11.5 shows the procedure which relies on

Table 11.1: Information exchanged between the EFS and OCS for implementing a follow-me migration procedure.

Ref. Point	Source	Destination	Information	Action	Step
E2	EFS service platform	EFS function manager	Resource ID, Wi-Fi station IDs, Wi-Fi signal level	Consume EFS Services related to the Wi-Fi information of surrounding Wi-Fi stations	CR.0
O3	EFS function manager	EFS stack orchestrator	Function instance ID, Dst resource ID	Request the migration of the function ID to the target resource ID	CR.3
	EFS stack orchestrator	EFS function manager	Migration status	Feedback on the requested migration	CR.6
Oo1	EFS stack orchestrator	EFS resource orchestrator	Function instance ID, Src resource ID, Dst resource ID	Request the migration of the function ID from Src resource ID to Dst resource ID	CR.4
	EFS resource orchestrator	EFS stack orchestrator	Migration status	Feedback on the requested migration	CR.6
O4	EFS resource orchestrator	VIM	Function instance ID, Src resource ID, Dst resource ID	Request the migration of the function ID from Src resource ID to Dst resource ID	CR.5
	VIM	EFS Resource Orchestrator	Function instance ID, Src resource ID, Dst resource ID	Feedback on the requested migration	CR.6

an EFS service providing measurements and information regarding the signal level as seen by all the Wi-Fi-capable EFS resources. Such EFS service is capable of providing the signal level of individual Wi-Fi stations as received at the virtual access point. The procedure of the measurement is the following, as also illustrated in Figure 11.5 and summarised in Table 11.1:

- **(A.1)** A dedicated EFS application (i.e., Wi-Fi mon in Figure 11.5) runs on every Wi-Fi-capable EFS resource and perform the corresponding measurements on the signal level.
- **(A.2)** The Wi-Fi monitoring application publishes the signal level measurements via an EFS service through the EFS service platform. The involved reference point is E2.

The OCS procedure for the migration of the virtual AP based on Wi-Fi signal level is the following as also illustrated in Figure 11.5:

- **(CR.0)** The EFS function manager associated to the virtual access point periodically consumes the EFS service providing the Wi-Fi signal level as seen from the EFS resources. The involved reference point is E2.
- **(CR.1)** Based on this information, the EFS function manager monitors the coarse location of the robots. This is a step internal to theEFS function manager.
- **(CR.2)** Based on the coarse location of the robot, the EFS function manager decides when a migration of the virtual access point is needed (e.g., the robots are closer to a given EFS resource than the one they are currently connected to). This is a step internal to the EFS function manager.
- **(CR.3)** TheEFS function manager contacts the EFS stack orchestrator to request the migration of the EFS function. The involved reference point is O3.
- **(CR.4)** The EFS stack orchestrator then contacts the EFS resource orchestrator for allocating the required resources (e.g., CPU, RAM, storage) on the target EFS Resource. The involved reference point is Oo1.

- **(CR.5)** If the migration request can be satisfied, the EFS resource orchestrator instructs the VIM to migrate the virtual access point to the target EFS resource. The involved reference point is O4.
- **(CR.6)** Feedback is provided to all the OCS components on the result of the procedure (e.g., successful or not). The involved reference points are O4, Oo1, and O3.

Finally, Table 11.1 reports the details on the information exchanged via the various reference points in the different migration steps. For each reference point, the source and destination components are identified as well as the information and the associated action. While the reported information allows to implement a generic migration, it is worth highlighting that from an OCS point of view it is the consumption of context information (i.e., via the Wi-Fi monitoring EFS service) the makes the virtual access point following the robots movement. As a result, the context information serves as a trigger to the general-purpose migration procedure implemented by the OCS components.

12. Conclusions of Part Three

The third part of this thesis has identified the edge and fog as key pillars of future networks where intelligence and innovations will be increasingly applied. There is however not yet a common unified platform that integrates and federates these two pillars together. Whilst the edge is more infrastructure-oriented and hence easier to integrate, the fog tends to be more volatile with resources appearing and disappearing on the go, and belonging to different owners. The opportunities for such unified framework are clearly acknowledged, but there remains to be several challenges that need to be addressed first before such a common framework could emerge. These include:

1. The dynamic discovery of volatile and non-volatile resources;
2. The federation of these resources when they belong to different domains and owners;
3. The support of multi-tenancy in particular for the volatile fog resources;
4. The customisation and interworking of different virtualisation technologies suitable to each type of resources (i.e., edge and fog);
5. The dynamic placement of functions and applications across the continuum of fog and edge;
6. The automation and dynamic allocation and management of the resources;
7. The security, trust and privacy considerations.

The overcoming of these challenges would hence enable a convergent 5G multi-RAT access through the integrated virtualised edge and fog solution, which envisages two main components: the EFS and the OCS. The former provides a low latency integrated virtualised environment distributed across the fog and edge to support multi-RAT convergence. The latter instead leverages on extended Software Defined Networking (SDN), NFV, and MEC tools to build and maintain the EFS, by enabling the automatic integration and federation of EFS resources into a unified hosting environment, despite their heterogeneity, ownership, and volatility.

> Publications covering the design of an integrated edge and fog system for multi-access convergence and related concepts are [Kim+18], [L C+18d], [Rap+18], [Rez+17].

Finally, an exemplary use case, namely fog-assisted robotics, has been presented with the goal of showing the benefits of the proposed EFS and OCS architecture. To that end, this part has first enumerated the limitations of today's cloud robotics, that is cloud facilities traditionally reside far away from the robots and while the cloud providers can enforce SLA in their infrastructure, very little can be ensured in the network between the robots and the cloud. Therefore, it has presented the need to shift towards the edge and fog for robotics applications and services so as to offer the same advantages of the cloud but without its limitations. To that end, a set of EFS applications, functions, and services has been identified tailored to the offering of a robotics cleaning service and synchronised delivery in a shopping mall environment. At the end, an exemplary interaction between the EFS and OCS components has been presented showing how a follow-me migration

mechanisms can be implemented via the usage of context information.

> The publication [Ant+18] covers the design and experimental assessment of a fog-assisted robotics application showing the benefits of using context information.

Conclusions

13. Conclusions

5G networks will carry more traffic than their predecessor, and this traffic will exhibit disparate characteristics, imposing very different requirements and constraints on the network design. Current network architectures are very rigid and inflexible in terms of the way they manage users' traffic, and are not capable either of quickly deploying new services on demand to cope with the dynamic needs from the customers. Therefore, future network architectures should be characterised by an *enhanced flexibility*. Software Defined Networking (SDN) is seen as one of the key tool to provide this required flexibility. To that end, this thesis departed from the general SDN framework defined by the Open Networking Foundation (ONF) and fully designed a compatible architecture suitable for future network operators. Among the plethoras of use cases and foreseen services, Distributed Mobility Management (DMM) is considered a necessity in future mobile network deployments in order to offload the network core from traffic that can be locally routed close to the access. Different actors have been working on this area, being the Internet Engineering Task Force (IETF) a major venue where most of the solutions have been discussed so far, while 3rd Generation Partnership Project (3GPP) has more recently started to work on distributed mobility architectures. Although there have been many different proposals, most of them share a characteristic: they are an evolved version of current IP mobility based solutions. While these are enough to offload the network core, and pose no significant deployment concerns, operators are already looking into SDN-based DMM solutions since they can potentially reduce the complexity and costs incurred by service creation and network operation. Therefore, it is important to understand how an SDN-based solution might look like when providing DMM support.

One of the main contributions of this thesis is the analytic and experimental evaluation of two key DMM protocol families: IP mobility and SDN-based. While the Proxy Mobile IPv6 (PMIPv6)-based solution is available in literature, this thesis thoroughly designed, modelled, and implemented the SDN-based solution. Additionally, this thesis walked the path of decomposing the functions that a DMM solution should have and identify how these can be implemented in an DMM-based solution. Moreover, existing state-of-the-art solutions are not generally studied both analytically and experimentally as it is done in this thesis, thus providing solid insights on how to apply DMM concepts in future mobile networks. By implementing the proposed SDN architecture and testing it on a medium size test-bed, this thesis demonstrated how easy and quick would be for an operator to create and put into operation new services, like the proposed SDN-based DMM. The results obtained from analysis and experiments show that the performance of the analysed solutions depends on the scenario being considered, but also indicate that SDN approaches have a big potential: (*i*) achievable performance is good and even better than the one of the PMIPv6-based solution, (*ii*) the solution can be easily implemented, and (*iii*) provides additional flexibility in regards of how it behaves and provides service differentiation.

Publications covering the SDN framework and the SDN-based DMM contributions, including related concepts, are [Con+16], [GLB15], [L C+17a], [L C+18b], [Wan+15a].

An open source SDN-based DMM implementation called OpenFlow-DMM is available at `http://odmm.net/openflow/`.

Another major contribution of this thesis is the analysis and design of a SDN-based unified data plane architecture for 5G, namely crosshaul, based on two main components: (*i*) the Crosshaul Forwarding Element (XFE) and (*ii*) the Crosshaul Common Frame (XCF). The XFE is a multi-layer switch based on packet – Crosshaul Packet Forwarding Element (XPFE) – and circuit – Crosshaul Circuit Switching Element (XCSE) – switching elements. While backhaul traffic is usually transmitted over the packet switch network, Common Public Radio Interface (CPRI) and diverse fronthaul traffic with stringent timing constraints are transmitted over the circuit switch network due the tight bandwidth and latency requirements the interface imposes to the network, which makes this interface quite rigid and costly. Aligned with new radio functional splits under study, Next Generation Fronthaul Interface (NGFI) and enhanced CPRI (eCPRI) relax the requirements of today's fronthaul in order to reach a more scalable interface so cheaper transport technologies can be used. At this purpose, packet switching enables statistical multiplexing when the peak to average radio access traffic load in 5G is high enough. Unified forwarding is enabled by the XCF format that is common across the various types of traffic and the various link technologies in the network. As a consequence, the unified data plane enables a common management of the integrated network in a SDN fashion. Therefore, traffic requirements, and hence services, could be easily enforced onto the network by leveraging the integrated and harmonised view provided by the unified data plane. As a result, the network operational costs can be significantly reduced.

Publications covering the design of the crosshaul network and related concepts are [Cav+17], [Dei+17], [Dei+16], [L C+16], [L C+18e], [Li+17].

Patents covering the design of the crosshaul network and related concepts are [LMK18], [L C+17b], [L C+17c].

The standard contribution [Ber+16] covers the crosshaul network requirements and related concepts.

This thesis has also presented a characterisation of a 5G transport network and the expected traffic mixture of network slices. Several simulations have been performed to understand the role of queueing disciplines in different scenarios, such as urban, industrial, and rural. This characterisation is key for properly engineering operator's networks to support next 5G services and satisfy the very stringent and diverse needs intrinsic to each of them. It indeed provides powerful insight on the candidate nodes in the network where a given service should be provided in order to fulfil its traffic requirements. The results have been compared with the constraints of the traffic flows defined in 3GPP and criticality has been identified for the motion control traffic part of the Ultra-Reliable and Low Latency Communications (URLLC) slice. Jitter requirements for such flow are only satisfied when the traffic is terminated in the access ring and a strict priority with preemption queueing discipline is used. Regarding the other flows and slices, traffic requirements are fulfilled in a failure-free scenario where the protection ring in the access and aggregation is not activated.

Publications covering the characterisation of a crosshaul network and related concepts are [L C+18e], [Mar+18].

The open source simulator developed for characterising the crosshaul network has been published with the name of SimPype and it is available at: `https://simpype.readthedocs.io/en/latest/`.

Furthermore, this thesis has identified a gap between current SDN solutions and carrier grade network requirements under Operations, Administration and Maintenance (OAM) point of view. An analysis of widely-deployed OAM and SDN technologies has been hence performed showing that the stateless nature of OpenFlow poses significant scalability and accuracy problems in monitoring and managing the network. To overcome these issues, this thesis proposes an Adaptive Telemetry System (ATS) to enable locally on the switches active measurements (e.g., delay, bandwidth, etc.) and their reporting (e.g., alarms). The design approach chosen for ATS showed to provide compatibility with standard OpenFlow switches and controllers. An Application Programming Interface (API) has been defined for enabling the remote configuration of telemetry procedures, which adopt a Finite State Machine (FSM) implementation. This enables the switches to locally execute the stateful procedures required for active monitoring. Finally, an experimental evaluation has been presented, showing the benefits of ATS compared to legacy-SDN solutions. Particularly, ATS proved to bring significant benefits in terms of offloading the control plane, and the Network Controller (NC), as well as higher accuracy in the performed measurements, which comply with the performance requirements defined by 3GPP for 5G networks. To that end, the delay and bandwidth measurements obtained with ATS have proven to match the ones obtained with reference non-SDN tools, while providing higher flexibility in the type of measurements that could be performed. Moreover, ATS proved to be able to manage the periodical generation of messages over a large number of ports (up to 256) while running on a single CPU core. Finally, this thesis provided some implementation insights on ATS and some deployment considerations regarding the clock distribution in the network.

The publication [L C+18a] covers the design of an Adaptive Telemetry System (ATS) for 5G SDN-based transport networks.

The patent [Per+17b] covers OAM functionalities in SDN-based networks employing the OpenFlow Southbound Interface (SBI).

Next, this thesis has identified the edge and fog as key pillars of future networks where intelligence and innovations will be increasingly applied. There is however not yet a common unified platform that integrates and federates these two pillars together. Whilst the edge is more infrastructure-oriented and hence easier to integrate, the fog tends to be more volatile with resources appearing and disappearing on the go, and belonging to different owners. The opportunities for such unified framework are clearly acknowledged, but there remains to be several challenges that need to be addressed first before such a common framework could emerge. These include: (*i*) the dynamic discovery of volatile and non-volatile resources, (*ii*) the federation of these resources when they belong to different domains and owners, (*iii*) the support of multi-tenancy in particular for the volatile fog resources, (*iv*) the customisation and interworking of different virtualisation technologies suitable to each type of resources (i.e., edge and fog), (*v*) the dynamic placement of functions and applications across the continuum of fog and edge, (*vi*) the automation and dynamic allocation and management of the resources, and finally (*vii*) the security, trust and privacy considerations. The overcoming of these challenges would hence enable a convergent 5G multi-Radio Access Technology (RAT) access through the integrated virtualised edge and fog solution, which envisages two main components: the Edge and Fog computing System (EFS) and the Orchestration and Control System (OCS). The former provides a low latency integrated virtualised environment distributed across the fog and edge to support multi-RAT convergence. The latter instead leverages on extended SDN, Network Function Virtualisation (NFV), and Mobile Edge Computing (MEC) tools to build and maintain the EFS, by enabling the automatic integration and federation of EFS resources into a unified hosting environment, despite their heterogeneity, ownership, and volatility.

Publications covering the design of an integrated edge and fog system for multi-access convergence and related concepts are [Kim+18], [L C+18d], [Rap+18], [Rez+17].

Finally, this thesis has presented an exemplary use case, namely fog-assisted robotics, with the goal of showing the benefits of the proposed EFS and OCS architecture. To that end, this part has first enumerated the limitations of today's cloud robotics and then presented the need to shift towards the edge and fog for robotics applications and services. To that end, a set of EFS applications, functions, and services has been identified tailored to the offering of a robotics cleaning service and synchronised delivery in a shopping mall environment. At the end, an exemplary interaction between the EFS and OCS components has been presented showing how a follow-me migration mechanisms can be implemented via the usage of context information.

The publication [Ant+18] covers the design and experimental assessment of a fog-assisted robotics application showing the benefits of using context information.

14. Future work

Classical mobility approaches only focus on the User Equipment (UE) mobility while considering the network to be rigid and static. However, the edge and fog system exhibits a more fluid network scenario wherein the nodes providing connectivity and services to the UE are also moving. This requires the analysis and the design of new network-based mobility mechanisms that consider both parts (i.e., the UE and the network) as mobile.

> New network-based mobility protocols with fluid anchoring for supporting dynamic network environments.

Moreover, given the ever-increasing densification of the mobile networks, the UE is expected to experience a larger number of handovers than today. This aspect poses critical challenges in the design of mobility protocols and their corresponding management. Indeed, these protocols not only will need to support an increasing number of handover in the whole network, but they will also need to provide the necessary session continuity to those very-demanding 5G services, such as Ultra-Reliable and Low Latency Communications (URLLC).

> New handover mechanisms to achieve zero-downtime and session continuity for Ultra-Reliable and Low Latency Communications (URLLC).

Finally, the capability of today's devices to be simultaneously connected to multiple networks (e.g., Wi-Fi, 3G, LTE, 5G, etc.) thanks to the multiple radio interfaces, opens up a new degree of freedom for mobility mechanisms where distinct protocol stacks might be considered at once for achieving the desired mobility and session continuity.

> New cross-protocol mechanisms to achieve mobility and session continuity across multiple Radio Access Technology (RAT) at once or using one RAT to support the mobility of a second interface.

While this thesis has already presented simulation-based results on the capability of a crosshaul network to fulfil 5G services, those results relate to a static reference architecture and a failure-free scenario. As a natural next step, more extensive simulations are expected to consider distinct network topologies (e.g., rings, mesh, etc.) and switching techniques. This aspect gains considerable importance when considering an integrated edge and fog system where the network boundaries go beyond the Next Generation NodeB (gNB) encompassing the UE as integral part of the infrastructure.

> Analysis of the role of queueing disciplines and congestion avoidance mechanisms with respect to distinct network topologies as well as the UE being part of the network infrastructure.

Furthermore, extensive analysis on the impact of volatile links on traffic delay and jitter is expected

for the different network slices.

> Analysis of the impact of wireless transport links on the target packet-error rate for URLLC services.

This includes the design and analysis of path restoration mechanisms where part of the network is composed of both wireless transport and access links and various network segments (e.g., L2 connectivity *vs* IP connectivity).

> New congestion-aware path restoration mechanisms for heterogeneous traffic including delay- and jitter-sensitive services.

Future work may include the analysis and experimental assessment of the proposed edge and fog integrated architecture spanning both networking and computing domains. Within the scope of the networking domain, the presented Software Defined Networking (SDN) framework has already been assessed in a mid-size test-bed considering powerful Commercial Off-The-Shelf (COTS) devices. Nonetheless, the edge and the fog also consider resource-constrained devices characterised by mobility and volatility. The impact of current SDN technologies (e.g., OpenFlow) on this new category of devices is still to be investigated, notably on the control and data planes.

> Experimental assessment of the impact of the OpenFlow protocol on the performance of resource-constrained and battery-powered devices (e.g., battery lifetime).

It is expected that only a limited set of data plane functionalities can be provided with constrained resources, thus requiring a re-design of the SDN applications so as to encompass these devices as part of the networking infrastructure.

> Analysis and experimental assessment of SDN data plane functionalities on resource-constrained devices (e.g., supported protocols, required hardware/computational capabilities, power consumption, etc.).

For what concerns the control plane instead, the analysis may involve the communication mechanisms of the control plane which, in case of OpenFlow, is nowadays based on TCP and eventually Transport Layer Security (TLS). These transport protocols have already proved to be too demanding for very resource-constrained devices.

> Analysis and experimental assessment of SDN control plane functionalities on resource-constrained devices (e.g., security, connection-oriented *vs* connectionless, availability, etc.)

Within the scope of the computing domain, a future line of work is the analysis of the impact of different hypervisors and virtualisation technologies on the 5G services when provided in software.

> Analysis and experimental assessment of virtualisation technologies on mobile and resource-constrained devices (e.g., footprint, power consumption, performance impact, etc.).

Similar to the network-based mobility, the computing domain needs to evolve in order to encompass dynamic scenarios where the computing device moves and it is not statically placed in a data centre. This includes new mechanisms for the on-the-fly migration of virtual functions and applications and the corresponding optimisations to meet the target session continuity.

> New migration mechanisms for virtual functions and applications migration to achieve zero-downtime and session continuity for Ultra-Reliable and Low Latency Communications (URLLC) and different virtualisation technologies.

While the hypervisor supports the migration of the virtualised applications and functions, the underlying network need to be reconfigured accordingly to reroute the traffic. This concept extends the classical mobility approach where the UE is the only entity moving and the application is statically placed. Indeed, in an edge and fog system the applications are distributed on the physical

infrastructure which, compared to cloud, is not static but rather dynamic.

> Integration of migration mechanisms with network-mobility mechanisms in dynamic and volatile environments.

Moreover, new software paradigms may be needed so as to develop the software in an edge and fog-native way, which requires more flexibility and agility compared to today's cloud environments.

> New software-design paradigms based on micro-services which consider the distributed and unreliable nature of the environment as core assumption.

The distributed nature of the fog-native applications also requires a rethinking of the communication bus that interconnects the different software components. Indeed, nowadays message bus have been designed for cloud environments characterised by powerful resources and big reliable pipes interconnecting them.

> New message bus suitable to run on resource-constrained devices aimed at interconnecting the distinct components of a fog-native application as well as distributing and replicating its internal state.

Finally, a set of new techniques and tools are required to enable continuous integration, and more broadly DevOps, in edge and fog systems in addition to what is done nowadays in the cloud. This would require the possibility of testing the software over federated environments which needs to be properly defined first. Indeed, one of the biggest challenges to develop applications in a fog-native fashion, it is the capability of performing all the testing necessary to ensure the quality of the software. While the homogeneity of the cloud makes this task relatively easy, the heterogeneity of the edge and fog makes it significantly hard. Testing the application under any possible conditions and combination of events typical of real environments is therefore a challenge that needs to be addressed in order to deliver a carrier-grade edge and fog system.

> A testing environment and set of common procedures for providing a fog-native certification need to be defined. This may include the definition of distinct fog-native levels where the environment might be more or less dynamic and volatile.

To that end, the definition and design of federation mechanisms for heterogeneous edge and fog systems presents a large number of research challenges which have not been yet addressed nor solved.

> Analysis and design of federation mechanisms for edge and fog systems including technical (e.g., coexistence of virtualisation domains, federated monitoring, security, etc.), business (e.g., Service Level Agreement (SLA), pricing, smart contracts, automatic negotiations, etc.), and regulatory aspects (e.g., liability, responsibility, privacy, etc.).

Once the above challenges are tackled, the materialisation of the edge and fog system will enable the evolution of today's ad-hoc and monolithic robotics systems towards an infrastructure-less and on-demand robotics service offering.

> Analysis and design of robotics systems leveraging the on-demand computing and networking infrastructure available at the edge end fog, including the consumption of third-party context information.

Although the research on 5G seeded many futuristic ideas and concepts, like the ones presented in the last part of this thesis, it is already clear that many of them will not materialise in real 5G deployments. Indeed, while 5G standardisation is finalising, recent advances in machine learning, distributed ledger mechanisms, artificial intelligence, and Terahertz communications are paving the road for the coming of 6G. To that end, research community and industries are already looking to bring forward those ideas beyond 5G towards another next generation of mobile networks. Moreover, a paradigm shift towards truly intelligent societies and more dynamic value chains might

arise with the advent of 6G where not only vertical industries can benefit from the mobile network advancements but also the end users at large.

Appendix

A. SimPype: a simulation framework

Figure A.1: SimPype logo

SimPype is a simulation framework based on SimPy[1] that relies on the concepts of*resource* and *pipe*. SimPype decouples the resource from its queue (i.e., pipe) in such a way multiple queueing techniques can be used with the same resource. SimPype also allows to create both custom resource and pipe models that can be reused in multiple simulations. SimPype supports only Python >= 3.3. Previous versions of Python are not supported. The quickest way to install SimPype is via pip3:

```
1 $ pip3 install simpype
```

SimPype automatically installs SimPy as dependency. SimPype documentation can be found on ReadTheDocs[2] while the source code repository is available on GitHub[3].

Scope

SimPype is tailored to scenarios where the queueing disciplines and the resources occupation are key parts of the system under simulation. Telecommunication networks, people queueing at a post office, supermarket, car wash, cafeteria, etc. are examples of such scenarios.

Concept

A SimPype simulation environment comprises at least one generator and one resource which are connected via a pipeline. The generator generates messages with a given arrival time. Those messages are first enqueued in the resource pipe and next processed by the resources according with a service time. The simulation steps can be summarised as follows:

1. The generator waits a random arrival time and generates a message;
2. The generator sends the message to the resource;
3. The message is enqueued in the resource's pipe;

[1] Simpy: `https://simpy.readthedocs.io/en/latest/`
[2] SimPype documentation: `http://simpype.readthedocs.io/en/latest/`
[3] SimPype source code: `https://github.com/Mallets/SimPype`

4. When the resource becomes available, the message is dequeued from the pipe;
5. The message is served by the resource;
6. The message leaves the resource after a random service time and is sent to the next resource (if any) - Go to step 3.

Any simulation step can be customised as desired. SimPype also provides a built-in logging system for your simulation that automatically logs the simulation steps 3, 4, and 5. Please refer to Chapter A.2 for more information on the logging system.

A.1 A simple simulation

To build a SimPype simulation, you simply need a console environment and a text editor. Write the following block of code into a text file, e.g., simple.py.

```
# [Mandatory] Import SimPype module
import simpype
import random
# [Mandatory] Create a SimPype simulation object
sim = simpype.Simulation(id = 'simple')
# [Mandatory] Add at least one generator to the simulation
gen0 = sim.add_generator(id = 'gen0')
gen0.random['arrival'] = {
  0: lambda: random.expovariate(2.0)
}
# [Mandatory] Add at least one resource to the simulation
res0 = sim.add_resource(id = 'res0')
# [Mandatory] Assign a service time
res0.random['service'] = {
  0: lambda: random.expovariate(3.0)
}
# [Mandatory] Add a pipeline connecting the generator to the
#             resource
p0 = sim.add_pipeline(gen0, res0)
# [Mandatory] Run the simulation, e.g., until t=5
#             sim.run calls Simpy's env.run
#             Any args passed to sim.run is then passed to env.run
sim.run(until = 5)
```

Now run the simulation by typing the following command in the console:

```
$ python3 simple.py
```

SimPype automatically logs the simulation results and its format is described in Chapter A.2. Please note that SimPype does not provide any tools for parsing and visualising the log data. Nevertheless, this data format is well-suited for being directly processed by data manipulation tools like pandas[4] for python or tidyverse[5] for R.

A.2 Logging system

SimPype automatically logs the simulation results in a log directory having the following structure in case of the simulation presented in Chapter A.1:

```
<working directory>
|-- simple.py
|-- log
  |-- simple
```

[4] Pandas: `https://pandas.pydata.org/`
[5] Tidyverse: `https://www.tidyverse.org/`

```
5       |-- <simulation date i.e. '2017-06-05 10:31:30.512772'>
6         |-- sim.cfg
7         `-- sim.log
```

sim.cfg contains information about the simulation environment and has the following format:

```
1  Simulation Seed: 1369068917606710528
2  Simulation Time: 5.000000000
3  Execution Time: 0.003524998
```

The built-in system produces the logs in a tidy format where each variable is saved in its own column and each observation is saved in its own row. Therefore, sim.log contains the log of the simulation events and looks like:

```
1  timestamp,message,seq_num,resource,event
2  0.000000000,gen0,0,res0,pipe.in
3  0.000000000,gen0,0,res0,pipe.out
4  0.252555552,gen0,0,res0,resource.serve
5  0.722431377,gen0,1,res0,pipe.in
6  0.722431377,gen0,1,res0,pipe.out
7  0.869881996,gen0,1,res0,resource.serve
8  1.413266674,gen0,2,res0,pipe.in
9  1.413266674,gen0,2,res0,pipe.out
10 1.478382544,gen0,2,res0,resource.serve
11 2.833221707,gen0,3,res0,pipe.in
12 2.833221707,gen0,3,res0,pipe.out
13 3.117096444,gen0,3,res0,resource.serve
14 3.455033536,gen0,4,res0,pipe.in
15 3.455033536,gen0,4,res0,pipe.out
16 4.174690658,gen0,5,res0,pipe.in
17 4.301555284,gen0,6,res0,pipe.in
18 4.587560103,gen0,4,res0,resource.serve
19 4.587560103,gen0,5,res0,pipe.out
20 4.898210753,gen0,5,res0,resource.serve
21 4.898210753,gen0,6,res0,pipe.out
22 4.975600594,gen0,6,res0,resource.serve
```

Let's analyse the first three log entries:

```
1  timestamp,message,seq_num,resource,event
2  0.000000000,gen0,0,res0,pipe.in
```

A message with id gen0 and sequence number 0 has been enqueued to the pipe of resource res0 in the queue default at simulation time 0.000000000.

```
1  timestamp,message,seq_num,resource,event
2  0.000000000,gen0,0,res0,pipe.out
```

The message with id gen0 and sequence number 0 has been dequeued from the pipe of resource res0 and queue default at simulation time 0.000000000. This means that the resource was available as soon as the message reached the resource. Therefore, the time spent waiting in the pipe was 0.

```
1  timestamp,message,seq_num,resource,event
2  0.252555552,gen0,0,res0,resource.serve
```

The resource res0 served the message with id gen0 and sequence number 0 at simulation time 0.252555552.

Change log directory

You can change the default log directory by setting the following variable in the simulation environment:

```
1 import simpype
2 sim = simpype.Simulation(id = 'simple')
3 sim.log.dir = '<your preferred dir>'
```

Please make sure you have writing permissions to *<your preferred dir>*.

Log custom message properties

You can configure SimPype's logger to log any additional message properties as you wish by calling the following function in the simulation environment:

```
1 import simpype
2 sim = simpype.Simulation(id = 'simple')
3 sim.log.property('test')
4 gen0.message.property['test'] = {0: lambda: 1}
```

sim.log file now has a column containing the value of test message property:

```
1 timestamp,message,seq_num,resource,event,test
2 0.000000000,gen0,0,res0,pipe.in,1
```

If a message does not have the custom property, SimPype logs NA instead.

Print the logs

If you prefer to print the logs instead of storing them in a file, you can do it by setting the following variables in the simulation environment:

```
1 import simpype
2 sim = simpype.Simulation(id = 'simple')
3 sim.log.file = False
4 sim.log.print = True
```

A.3 Pipeline

Pipelines allow to arbitrarily interconnect generators and resources by chaining them. Messages belonging to a pipeline automatically flow from a resource to another without the need of explicitly defining the hops. Please note that a generator (and their messages) belong only to a single pipeline. If multiple pipelines are needed simultaneously, they need to be merged first. See Branching pipeline in the following for more details.

Single pipeline

Let's start with a simulation where messages are generated by Generator#0 and are served by Resource #0 and Resource #1.

```
1 |Generator #0| -> |Resource #0| -> |Resource #1|
```

The SimPype code would hence be:

```
1 import simpype
2 import random
3 sim = simpype.Simulation(id = 'single')
4 gen0 = sim.add_generator(id = 'gen0')
5 gen0.random['arrival'] = {0: lambda: random.expovariate(1.0)}
```

```
6  res0 = sim.add_resource(id = 'res0')
7  res0.random['service'] = {0: lambda: random.expovariate(2.0)}
8  res1 = sim.add_resource(id = 'res1')
9  res1.random['service'] = {0: lambda: random.expovariate(2.0)}
10 p0 = sim.add_pipeline(gen0, res0, res1)
11 sim.run(until = 5)
```

Overlapping pipelines

Now let's continue with a simulation scenarios like the following:

```
1  |Generator #0| -\                      /-> |Resource #1|
2                  )-> |Resource #0| -(
3  |Generator #1| -/                      \-> |Resource #2|
```

In this scenario we want to reproduce the following interconnection:

```
1  |Generator #0| -> |Resource #0| -> |Resource #1|
2
3  |Generator #1| -> |Resource #0| -> |Resource #2|
```

As it can be noticed, there are two distinct paths/pipelines that overlap at Resource#0. However, any messages generated by Generator #0 should end to Resource #1. Similarly, any messages generated by Generator #1 should end to Resource #2. In this scenario, Resource#0 is hence shared between the two pipelines. The SimPype code would hence be:

```
1  import simpype
2  import random
3  sim = simpype.Simulation(id = 'overlap')
4  gen0 = sim.add_generator(id = 'gen0')
5  gen0.random['arrival'] = {0: lambda: random.expovariate(1.0)}
6  gen1 = sim.add_generator(id = 'gen1')
7  gen1.random['arrival'] = {0: lambda: random.expovariate(1.0)}
8  res0 = sim.add_resource(id = 'res0')
9  res0.random['service'] = {0: lambda: random.expovariate(4.0)}
10 res1 = sim.add_resource(id = 'res1')
11 res1.random['service'] = {0: lambda: random.expovariate(2.0)}
12 res2 = sim.add_resource(id = 'res2')
13 res2.random['service'] = {0: lambda: random.expovariate(2.0)}
14 p0 = sim.add_pipeline(gen0, res0, res1)
15 p1 = sim.add_pipeline(gen1, res0, res2)
16 sim.run(until = 2.5)
```

Branching pipeline

Now let's continue with a pipeline having a branching point with one generator and three resources:

```
1                                      /-> |Resource\#1|
2  |Generator\#0| -> |Resource\#0| -(
3                                      \-> |Resource\#2|
```

There are two possible options at this stage:

1. Serve a copy of the same message to both Resource #1 and Resource #2;
2. Either serve a message to Resource#1 or to Resource #2. Please refer to Chapter A.5 to understand how the next hop of the messages can be dynamically changed.

In case of serving a copy of the same message to both Resource #1 and Resource #2, the SimPype code would hence be:

```
import simpype
import random
sim = simpype.Simulation(id = 'single')
gen0 = sim.add_generator(id = 'gen0')
gen0.random['arrival'] = {0: lambda: random.expovariate(1.0)}
res0 = sim.add_resource(id = 'res0')
res0.random['service'] = {0: lambda: random.expovariate(2.0)}
res1 = sim.add_resource(id = 'res1')
res1.random['service'] = {0: lambda: random.expovariate(2.0)}
res2 = sim.add_resource(id = 'res2')
res2.random['service'] = {0: lambda: random.expovariate(2.0)}
p0 = sim.add_pipeline(gen0, res0, res1)
p1 = sim.add_pipeline(gen0, res0, res2)
pM = sim.merge_pipeline(p0, p1)
sim.run(until = 5)
```

Please note the use of merge_pipeline(). This function merges multiple pipelines into a single one, thus creating the branching point. Without calling the merge_pipeline() function, the only active pipeline would have been p1.

A.4 Random variables

SimPype comes with a custom random variable generation system that allows you to generate random values according to different random distributions depending on the current simulation time.

```
import simpype
sim = simpype.Simulation(id = 'test')
myrand = simpype.Random(sim, {
  initial_time : lambda_function
  ...
})
```

Where each dictionary element is so defined:

- *initial_time* is the element key and must be of *int* or *float* type. It represents the initial simulation time at which the *lambda_function* is invoked;
- *lambda_function* is the element value. It is mandatory that for the value to be a lambda function. Such function must return a value, usually a *int* or a *float*;

An example of random variable initialisation is the following:

```
import simpype
import random
sim = simpype.Simulation(id = 'test')
myrand = simpype.Random(sim, {
  # From t=0 to t=10, the random variable returns
  # the constant value of 3.0
  0 : lambda: 3.0,
  # From t=10 to t=20, the random variable returns
  # value uniformly distributed between 2.5 and 3.5
  10: lambda: random.uniform(2.5, 3.5),
  # From t=20 to t=inf, the random variable returns
  # a value exponentially distributed with lambda 0.20
  20: lambda: random.expovariate(0.20)
})
```

To generate a random value:

```
# Simulation time = 5.0
random_value = myrand.value     # random_value = 3.0
```

```
...
# Simulation time = 15.0
random_value = myrand.value     # random_value = 3.2476115513945767
...
# Simulation time = 25.0
random_value = myrand.value     # random_value = 7.374759019459148
```

As it can be noticed, depending on the current simulation myrand.value returns a random value according to a different random distribution.

Generator arrival time

The arrival time of a generator is described with a Random variable.

```
import simpype
import random
sim = simpype.Simulation(id = 'simple')
gen0 = sim.add_generator(id = 'gen0')
# Start generating events at a random simulation time
gen0.random['arrival'] = {
  # From t=0 to t=10, the arrival time is constant to 3.0
  0 : lambda: 3.0,
  # From t=10 to t=20, the arrival time is uniformly distributed
  # between 2.5 and 3.5
  10: lambda: random.uniform(2.5, 3.5),
  # From t=20 to t=inf, the arrival time is exponentially
  # distributed with lambda 0.20
  20: lambda: random.expovariate(0.20)
}
```

Please note that in this case there is no need of calling the simpype.Random constructor. The generator object automatically converts the dictionary into a Random object.

Resource service time

The service time of a resource is described with a Random variable.

```
import simpype
import random
sim = simpype.Simulation(id = 'simple')
res0 = sim.add_resource(id = 'res0')
res0.random['arrival'] = {
  # From t=0 to t=10, the service time is constant to 3.0
  0 : lambda: 3.0,
  # From t=10 to t=20, the service time is uniformly distributed
  # between 2.5 and 3.5
  10: lambda: random.uniform(2.5, 3.5),
  # From t=20 to t=inf, the service time is exponentially
  # distributed with lambda 0.20
  20: lambda: random.expovariate(0.20)
}
```

Please note that in this case there is no need of calling the simpype.Random constructor. The resource object automatically converts the dictionary into a Random object.

Message property

A message property can be described with a a Random variable.

```
import simpype
import random
sim = simpype.Simulation(id = 'simple')
gen0 = sim.add_generator(id = 'gen0')
```

```
5  gen0.message.property['test'] = {
6    # Every message generated between t=0 and t=10 will have the
7    # 'test' property value equal to 3.0
8    0 : lambda: 3.0,
9    # Every message generated between t=10 and t=20 will have the
10   # 'test' property uniformly distributed between 2.5 and 3.5
11   10: lambda: random.uniform(2.5, 3.5),
12   # Every message generated between t=20 and t=inf will have the
13   # 'test' property exponentially distributed with lambda 0.20
14   20: lambda: random.expovariate(0.20)
15 }
```

Please note that in this case there is no need of calling the simpype.Random constructor. The message object automatically converts the dictionary into a Random object. Please also note that property values can be randomly generated. Nevertheless, once they are generated, they will always return the same value unless an explicit refresh is called:

```
1  message.property['test'].refresh()
```

A.5 Message

Messages are the units processed by the resources and can store arbitrary information, called properties in SimPype. Message properties can be of any values, including Random objects (see Chapter A.4).

```
1  import simpype
2  import random
3  sim = simpype.Simulation(id = 'simple')
4  gen0 = sim.add_generator(id = 'gen0')
5  gen0.message.property['rand_prop'] = {
6    # Every message generated between t=0 and t=10 will have
7    # the 'test' property value equal to 3.0
8    0 : lambda: 3.0,
9    # Every message generated between t=10 and t=20 will have
10   # the 'test' property uniformly distributed between 2.5 and 3.5
11   10: lambda: random.uniform(2.5, 3.5),
12   # Every message generated between t=20 and t=inf will have
13   # the 'test' property exponentially distributed with lambda 0.20
14   20: lambda: random.expovariate(0.20)
15 }
16 # Store the property as normal dictionary if
17 # no lambda function is present
18 gen0.message.property['dict_prop'] = {
19   'a': 'avalue',
20   'b': 'bvalue',
21 }
22 gen0.message.property['str_prop'] = 'mystr'
23 gen0.message.property['int_prop'] = 3
24 gen0.message.property['float_prop'] = 3.0
25 # You can also store objects
26 e = sim.env.event()
27 gen0.message.property['event_prop'] = e
```

Please note that in this case there is no need of calling the simpype.Random constructor. The message object automatically converts the dictionary into a Random object. Please also note that property values can be randomly generated, nevertheless once they are generated they will always return the same value unless an explicit refresh is called:

```
message.property['test'].refresh()
```

Drop

A message can be suddenly dropped by calling the function drop():

```
message.drop(id = 'bad luck')
```

In addition, a message can be dropped upon the occurence of a given event:

```
# Create a SimPy event
e = sim.env.event()
# Subscribe the dropping of the message to the event e
message.drop(id = 'event bad luck', event = e)
# Trigger the event
e.succeed()
# The message has now been dropped
```

The message is dropped only when the event e is triggered, that is succeed in SimPy notation.

Lifetime

A lifetime can be assigned to generated messages in the following way:

```
import simpype
import random
sim = simpype.Simulation(id = 'simple')
gen0 = sim.add_generator(id = 'gen0')
gen0.message.property['lifetime'] = {
  0: lambda: random.expovariate(0.20)
}
```

The message is dropped when the lifetime expires. To remove any lifetime from the message, use the following function:

```
message.unsubscribe(id = 'lifetime')
```

Event subscription

A message can be subscribed to a given event and a custom function can be executed upong event triggering, e.g.:

```
import simpype
import random
sim = simpype.Simulation(id = 'simple')
gen0 = sim.add_generator(id = 'gen0')
res0 = sim.add_resource(id = 'res0')
res1 = sim.add_resource(id = 'res1')

e = sim.env.event()
def c(message, value):
  # Value of the event, e.g. 'OK'
  message.property['myevent'] = value

@simpype.resource.service(res0)
def service(self, message):
  global e
  # Trigger the event
  e.succeed(value = 'OK')
  e = sim.env.event()
```

```
19
20 @simpype.resource.service(res1)
21 def service(self, message):
22   # Unsubscribe from the event
23   message.unsubscribe(id = 'mysub')
24
25 gen0.message.subscribe(event = e, callback = c, id = 'mysub')
```

The callback function must be defined according to the following format:

```
1 def callback(message, value):
2 ... your code here ...
```

Next hop

Let's assume we have a simulation scenario like the following:

```
1                                  /-> |Resource #0|
2 |Generator #0| -> |Splitter| -(
3                                  \-> |Resource #1| -> |Resource #2|
```

Messages can be either go to Resource #0 or to Resource #1 depending on Splitter decision. In this example, messages with even sequence number are sent to Resource #0 while messages with odd sequence number are sent to Resource #1 and next to Resource #2. To achieve this, the next hop of a message can be dynamically changed by setting the message *next* variable. *text* admits both Resource and Pipeline objects as values.

```
1  import simpype
2  import random
3  sim = simpype.Simulation(id = 'next')
4  gen0 = sim.add_generator(id = 'gen')
5  gen0.random['arrival'] = {0: lambda: 1.0}
6  res0 = sim.add_resource(id = 'res0')
7  res1 = sim.add_resource(id = 'res1')
8  res2 = sim.add_resource(id = 'res2')
9  splitter = sim.add_resource(id = 'splitter')
10 p0 = sim.add_pipeline(gen0, splitter)
11 p1a = sim.add_pipeline(splitter, res0)
12 p1b = sim.add_pipeline(res1, res2)
13 p2 = sim.add_pipeline(splitter, p1b)
14 pM = sim.merge_pipeline(p0, p1a, p1b, p2)
15
16 # Change next
17 @simpype.resource.service(splitter)
18 def service(self, message):
19 yield self.env.timeout(1.0)
20 if message.seq_num % 2 == 0:
21   message.next = res0
22 else:
23   message.next = p1b
24
25 sim.run(until = 10)
```

A.6 Resource

The resource behaviour can be customised in case of more complex operations are needed in addition to the simple random generated service time. To that end, the decorator *@simpype.resource.service* can be used. There are two ways of customising the behaviour of your resource:

Work directly on the simulation scenario and perform an in-line customisation; Create a resource model to be included in the simulation scenario. Either approaches are valid, however in-line customisation is more suited for small customisations while resource model is more suited for larger customisations and code re-usability (the same model can be imported multiple times in different simulations).

In-line customisation

In this example, the service time of the resource also depends on the message property value wait.

```
1  import simpype
2  import random
3  sim = simpype.Simulation(id = 'simple')
4  gen0 = sim.add_generator(id = 'gen0')
5  gen0.message.property['wait'] = {
6    0: lambda: random.uniform(0,1)
7  }
8  res0 = sim.add_resource(id = 'res0')
9  res0.random['service'] = {
10   0: lambda: 2.0
11 }
12
13 @simpype.resource.service(res0)
14 def service(self, message):
15   # Wait for a random time
16   yield self.env.timeout(self.random['service'])
17   # Wait for a time as reported in the message property
18   yield self.env.timeout(message.property['wait'].value)
19
20 sim.run(until = 10)
```

Custom model

Alternatively, a separate resource model can be created to implement the same resource behaviour: Edit *myresource.py* with a text editor and create a resource model as follows:

```
1  import simpype
2
3  class MyResource(simpype.Resource):
4  def __init__(self, sim, id, capacity = 1, pipe = None):
5    super().__init__(sim, id, capacity, pipe)
6
7    @simpype.resource.service
8    def service(self, message):
9      # Wait for a random time
10     yield self.env.timeout(self.random['service'])
11     # Wait for a time as reported in the message property
12     yield self.env.timeout(message.property['wait'].value)
13
14 # Do NOT remove. This is required for SimPype to build your model.
15 resource = lambda *args: MyResource(*args)
```

Create your simulation scenario including the new model:

```
1  import simpype
2  import random
3  sim = simpype.Simulation(id = 'simple')
4  gen0 = sim.add_generator(id = 'gen0')
5  gen0.message.property['wait'] = {
6    0: lambda: random.uniform(0,1)
7  }
8  res0 = sim.add_resource(id = 'res0', model = 'myresource')
```

```
res0.random['service'] = {
  0: lambda: 2.0
}
sim.run(until = 10)
```

Make sure that the file and directory structure is the following:

```
<working directory>
|-- simple.py
|-- myresource.py
```

If you want to change the directory where SimPype looks for custom models, set the following variable in the simulation environment:

```
import simpype
sim = simpype.Simulation(id = 'simple')
sim.model.dir = '<your model dir>'
```

Please make sure you have reading permissions for <*your model dir*>. In this case, the file and directory structure would look like:

```
<working directory>
|-- simple.py
<your model dir>
|-- myresource.py
```

A.7 Pipe

The pipe behaviour of a given resource can be customised in case of more complex queueing operations are needed in addition to the simple FIFO discipline. To that end, multiple queues can be added to the pipe and two decorators can be used to determine the enqueueing and dequeueing behaviour of the pipe. The two decorators are *@simpype.pipe.enqueue* and *@simpype.pipe.dequeue*. There are two ways of customising the behaviour of a pipe:

- Work directly on the simulation scenario and perform an in-line customisation;
- Create a *pipe* model to be included in the simulation scenario.

Either approaches are valid, however in-line customisation is more suited for small customisations while pipe model is more suited for larger customisations and code re-usability (the same model can be imported multiple times in different simulations).

In-line customisation

In this example, a priority queue with two service classes is implemented.

```
import simpype
import random
sim = simpype.Simulation(id = 'simple')
gen0 = sim.add_generator(id = 'gen0')
gen0.message.property['priority'] = {
  0: lambda: random.randint(0,1)
}
res0 = sim.add_resource(id = 'res0')
res0.pipe.add_queue('slow')
res0.pipe.add_queue('fast')
res0.random['service'] = {
  0: lambda: 2.0
}

@simpype.pipe.enqueue(res0.pipe)
```

```
16 def enqueue(self, message):
17   if message.property['priority'] == 0:
18     return self.queue['slow'].push(message)
19   elif message.property['priority'] == 1:
20     return self.queue['fast'].push(message)
21   else:
22     return message.drop('unsupported priority')
23
24 @simpype.pipe.dequeue(res0.pipe)
25 def dequeue(self):
26   if len(self.queue['fast']) > 0:
27     return self.queue['fast'].pop()
28   elif len(self.queue['slow']) > 0:
29     return self.queue['slow'].pop()
30   else
31     return None
32
33 sim.run(until = 10)
```

Custom model

Alternatively, a separate *pipe* model can be created to implement the same *pipe* behaviour. Edit *mypipe.py* with a text editor and create a *pipe* model as follows:

```
1  import simpype
2
3  class MyPipe(simpype.Pipe):
4    def __init__(self, sim, resource, id):
5      super().__init__(sim, resource, id)
6      self.add_queue(id = 'slow')
7      self.add_queue(id = 'fast')
8
9    @simpype.pipe.enqueue
10   def enqueue(self, message):
11     if message.property['priority'] == 0:
12       return self.queue['slow'].push(message)
13     elif message.property['priority'] == 1:
14       return self.queue['fast'].push(message)
15     else:
16       return message.drop('unsupported priority')
17
18   @simpype.pipe.dequeue
19   def dequeue(self):
20     if len(self.queue['fast']) > 0:
21       return self.queue['fast'].pop()
22     elif len(self.queue['slow']) > 0:
23       return self.queue['slow'].pop()
24     else
25       return None
26
27 # Do NOT remove. This is required for SimPype to build your model.
28 pipe = lambda *args: MyPipe(*args)
```

Create the simulation scenario including the new model:

```
1 import simpype
2 import random
3 sim = simpype.Simulation(id = 'simple')
4 gen0 = sim.add_generator(id = 'gen0')
5 gen0.message.property['priority'] = {
6   0: lambda: random.randint(0,1)
7 }
```

```
res0 = sim.add_resource(id = 'res0', pipe = 'mypipe')
res0.random['service'] = {
  0: lambda: 2.0
}
sim.run(until = 10)
```

Make sure that the file and directory structure is the following:

```
<working directory>
|-- simple.py
|-- mypipe.py
```

If you want to change the directory where SimPype looks for custom models, set the following variable in the simulation environment:

```
import simpype
sim = simpype.Simulation(id = 'simple')
sim.model.dir = '<your model dir>'
```

Please make sure you have reading permissions for <*your model dir*>. In this case, the file and directory structure would look like:

```
<working directory>
|-- simple.py
<your model dir>
|-- mypipe.py
```

A.8 Queue

The queue behaviour of a given pipe can be customised in case of more complex queueing operations are needed in addition to the simple FIFO buffer. To that end, two decorators can be used to determine the push and pop behaviour of the queue. The two decorators are *@simpype.queue.push* and *@simpype.queue.pop*. There are two ways of customising the behaviour of a queue:

- Work directly on the simulation scenario and perform an in-line customisation;
- Create a *queue* model to be included in the simulation scenario through a custom *pipe* model.

Either approaches are valid, however in-line customisation is more suited for small customisations while queue model is more suited for larger customisation and code re-usability (the same model can be imported multiple times in different simulations).

In-line customisation

In this example, a LIFO discipline is implemented.

```
import simpype
import random
sim = simpype.Simulation(id = 'simple')
gen0 = sim.add_generator(id = 'gen0')
gen0.message.property['priority'] = {
  0: lambda: random.randint(0,1)
}
res0 = sim.add_resource(id = 'res0')
res0.pipe.add_queue(id = 'lifo')
res0.random['service'] = {
  0: lambda: 2.0
}

@simpype.pipe.enqueue(res0.pipe)
def enqueue(self, message):
```

```
16    return self.queue['lifo'].push(message)
17
18  @simpype.pipe.dequeue(res0.pipe)
19  def dequeue(self):
20    return self.queue['lifo'].pop()
21
22  @simpype.queue.push(res0.pipe.queue['lifo'])
23  def push(self, message):
24    return self.buffer.append(message)
25
26  @simpype.queue.pop(res0.pipe.queue['lifo'])
27  def pop(self):
28    return self.buffer.pop(-1)
29
30  sim.run(until = 10)
```

Custom model

Alternatively, a separate pipe model and queue model can be created to implement the same discipline. Edit *mylifo.py* with a text editor and create a *pipe* model as follows:

```
1   import simpype
2
3   class MyQueue(simpype.Queue):
4     def __init__(self, sim, pipe, id):
5       super().__init__(sim, pipe, id)
6
7     @simpype.queue.push
8     def push(self, message):
9       return self.buffer.append(message)
10
11    @simpype.queue.pop
12    def pop(self):
13      return self.buffer.pop(-1)
14
15  class MyPipe(simpype.Pipe):
16    def __init__(self, sim, resource, id):
17      super().__init__(sim, resource, id)
18      self.add_queue(id = 'lifo', model = 'mylifo')
19
20    @simpype.pipe.enqueue
21    def enqueue(self, message):
22      return self.queue['lifo'].push(message)
23
24    @simpype.pipe.dequeue
25    def dequeue(self):
26      return self.queue['lifo'].pop()
27
28  # Do NOT remove. This is required for SimPype to build your model.
29  queue = lambda *args: MyQueue(*args)
30  pipe = lambda *args: MyPipe(*args)
```

Create your simulation scenario including the new model:

```
1   import simpype
2   import random
3   sim = simpype.Simulation(id = 'simple')
4   gen0 = sim.add_generator(id = 'gen0')
5   gen0.message.property['priority'] = {
6     0: lambda: random.randint(0,1)
7   }
8   res0 = sim.add_resource(id = 'res0', pipe = 'mylifo')
```

```
 9  res0.random['service'] = {
10    0: lambda: 2.0
11  }
12  sim.run(until = 10)
```

Make sure that the file and directory structure is the following:

```
1  <working directory>
2  |-- simple.py
3  |-- mylifo.py
```

If you want to change the directory where SimPype looks for custom models, set the following variable in the simulation environment:

```
1  import simpype
2  sim = simpype.Simulation(id = 'simple')
3  sim.model.dir = '<your model dir>'
```

Please make sure you have reading permissions for <*your model dir*>. In this case, the file and directory structure would look like:

```
1  <working directory>
2  |-- simple.py
3  <your model dir>
4  |-- mylifo.py
```

A.9 A simulation with priority queues

This simulation models a boarding gate for a flight with three separate classes: first, business, and economy:

```
1  |First class|-----\
2  |Business class|---+-> |Boarding gate|
3  |Economy class|---/
```

Boarding priority is given to the different classes according to the following order:

1. first class
2. business class
3. economy class

With first class having the highest priority and economy class having the lowest.

```
1  import simpype
2  import random
3  sim = simpype.Simulation(id = 'boarding')
4  # Create a generator
5  first = sim.add_generator(id = 'first')
6  first.to_send = 12
7  # Assign an arrival time
8  first.random['arrival'] = {
9    0    : lambda: random.expovariate(1.0 / 900),
10   1800: lambda: random.expovariate(1.0 / 30)
11 }
12 # Create a generator
13 business = sim.add_generator(id = 'business')
14 business.to_send = 24
15 # Assign an arrival time
16 business.random['arrival'] = {
17   0    : lambda: random.expovariate(1.0 / 450),
```

```
18    900 : lambda: random.expovariate(1.0 / 60),
19    1800: lambda: random.expovariate(1.0 / 30),
20  }
21  # Create a generator
22  economy = sim.add_generator(id = 'economy')
23  economy.to_send = 160
24  # Assign an arrival time
25  economy.random['arrival'] = {
26    0    : lambda: random.expovariate(1.0 / 60),
27    600  : lambda: random.expovariate(1.0 / 30),
28    1200: lambda: random.expovariate(1.0 / 10),
29  }
30  # Add a resource
31  gate = sim.add_resource(id = 'gate', pipe = 'p_priority')
32  gate.random['service'] = {
33    # The gate opens 30 mins from the simulation start
34    # Check boarding pass and passport takes ~10s
35    1800: lambda: random.expovariate(1.0 / 10)
36  }
37  # Add a pipeline connecting the generator to the resource
38  p0 = sim.add_pipeline(first, gate)
39  p1 = sim.add_pipeline(business, gate)
40  p2 = sim.add_pipeline(economy, gate)
41  # Run the simulation
42  sim.run()
```

Where the pipe model *p_priority* is so implemented:

```
1   import simpype
2
3   class Priority(simpype.pipe.Pipe):
4     def __init__(self, sim, resource, id):
5       super().__init__(sim, resource, id)
6       self.add_queue(id = 'express')
7       self.add_queue(id = 'fast')
8       self.add_queue(id = 'slow')
9
10    @simpype.pipe.dequeue
11    def dequeue(self):
12      if len(self.queue['express']) > 0:
13        m = self.queue['express'].pop()
14      elif len(self.queue['fast']) > 0:
15        m = self.queue['fast'].pop()
16      else:
17        m = self.queue['slow'].pop()
18      return m
19
20    @simpype.pipe.enqueue
21    def enqueue(self, message):
22      if message.id == 'first':
23        m = self.queue['express'].push(message)
24      elif message.id == 'business':
25        m = self.queue['fast'].push(message)
26      elif message.id == 'economy':
27        m = self.queue['slow'].push(message)
28      else:
29        m = self.queue['slow'].push(message)
30      return m
31
32  # Do NOT remove
33  pipe = lambda *args: Priority(*args)
```

Bibliography

References

Articles

[Agy+14] P. K. Agyapong et al. 'Design considerations for a 5G network architecture'. In: *IEEE Communications Magazine* 52.11 (November 2014), pages 65–75. ISSN: 0163-6804. DOI: 10.1109/MCOM.2014.6957145 (cited on page 57).

[Ahm+16] I. Ahmad et al. 'New concepts for traffic, resource and mobility management in software-defined mobile networks'. In: *2016 12th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*. January 2016, pages 1–8 (cited on page 60).

[Ant+18] K. Antevski et al. 'Enhancing Edge robotics through the use of context information'. In: *Workshop on Experimentation and Measurements in 5G (EM-5G'18)* (December 2018), pages 1–5. DOI: 10.1145/3286680.3286682 (cited on pages 16, 164, 170).

[ABT14] M. R. M. Assis, L. F. Bittencourt and R. Tolosana-Calasanz. 'Cloud Federation: Characterisation and Conceptual Model'. In: *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*. UCC '14. IEEE Computer Society, 2014, pages 585–590. ISBN: 978-1-4799-7881-6. DOI: 10.1109/UCC.2014.90 (cited on pages 140, 141).

[Bar+13] M. F. Bari et al. 'PolicyCop: An Autonomic QoS Policy Enforcement Framework for Software Defined Networks'. In: *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*. November 2013, pages 1–7. DOI: 10.1109/SDN4FNS.2013.6702548 (cited on page 118).

[BZO15] C. J. Bernardos, J. C. Zúñiga and P. O'Hanlon. 'Wi-Fi internet connectivity and privacy: Hiding your tracks on the wireless Internet'. In: *2015 IEEE Conference on Standards for Communications and Networking (CSCN)*. October 2015, pages 193–198. DOI: 10.1109/CSCN.2015.7390443 (cited on page 57).

[Ber+14] C. J. Bernardos et al. 'An architecture for software defined wireless networking'. In: *IEEE Wireless Communications* 21.3 (June 2014), pages 52–61. ISSN: 1536-1284. DOI: 10.1109/MWC.2014.6845049 (cited on pages 69, 91, 117).

[Bia+14] G. Bianchi et al. 'OpenState: Programming Platform-independent Stateful Openflow Applications Inside the Switch'. In: *SIGCOMM Comput. Commun. Rev.* 44.2 (April 2014), pages 44–51. ISSN: 0146-4833. DOI: 10.1145/2602204.2602211 (cited on page 121).

[Bif+16] R. Bifulco et al. 'Improving SDN with InSPired Switches'. In: *Proceedings of the Symposium on SDN Research*. SOSR '16. Santa Clara, CA, USA: ACM, 2016, 11:1–11:12. ISBN: 978-1-4503-4211-7. DOI: 10.1145/2890955.2890962 (cited on page 122).

[BR14] R. Bolla and M. Repetto. 'A Comprehensive Tutorial for Mobility Management in Data Networks'. In: *IEEE Communications Surveys Tutorials* 16.2 (February 2014), pages 812–833. ISSN: 1553-877X. DOI: 10.1109/SURV.2013.071913.00140 (cited on page 67).

[Cas+] C. Cascone et al. 'Fast failure detection and recovery in SDN with stateful data plane'. In: *International Journal of Network Management* 27.2 (), e1957. DOI: 10.1002/nem.1957 (cited on page 122).

[Cav+17] F. Cavaliere et al. 'Towards a unified fronthaul-backhaul data plane for 5G The 5G-Crosshaul project approach'. In: *Computer Standards & Interfaces* 51 (2017), pages 56–62. ISSN: 0920-5489. DOI: 10.1016/j.csi.2016.11.005 (cited on pages 15, 135, 168).

[CH13] C. Chaudet and Y. Haddad. 'Wireless Software Defined Networks: Challenges and opportunities'. In: *2013 IEEE International Conference on Microwaves, Communications, Antennas and Electronic Systems (COMCAS 2013)*. October 2013, pages 1–5. DOI: 10.1109/COMCAS.2013.6685237 (cited on page 69).

[CZ16] M. Chiang and T. Zhang. 'Fog and IoT: An Overview of Research Opportunities'. In: *IEEE Internet of Things Journal* 3.6 (December 2016), pages 854–864. ISSN: 2327-4662. DOI: 10.1109/JIOT.2016.2584538 (cited on pages 139, 140).

[Con+16] L. M. Contreras et al. 'Software-Defined Mobility Management: Architecture Proposal and Future Directions'. In: *Mobile Networks and Applications* 21.2 (April 2016), pages 226–236. ISSN: 1572-8153. DOI: 10.1007/s11036-015-0663-7. URL: https://link.springer.com/article/10.1007/s11036-015-0663-7 (Retrieved: 10th January 2019) (cited on pages 10, 96, 168).

[DMS12] E. Danna, S. Mandal and A. Singh. 'A practical algorithm for balancing the max-min fairness and throughput objectives in traffic engineering'. In: *2012 Proceedings IEEE INFOCOM*. March 2012, pages 846–854. DOI: 10.1109/INFCOM.2012.6195833 (cited on page 65).

[Dat+14] P. T. Dat et al. 'High-Capacity Wireless Backhaul Network Using Seamless Convergence of Radio-over-Fiber and 90-GHz Millimeter-Wave'. In: *Journal of Lightwave Technology* 32.20 (October 2014), pages 3910–3923. ISSN: 0733-8724. DOI: 10.1109/JLT.2014.2315800 (cited on page 102).

[Dei+17] T. Deiß et al. 'Dataplane Measurements on a Fronthaul (FH) and Backhaul (BH) integrated network'. In: *5th International Workshop on Cloud Technologies and Energy Efficiency in Mobile Communication Networks (CLEEN 2017)*. June 2017. URL: http://eprints.networks.imdea.org/1612/ (Retrieved: 10th January 2019) (cited on pages 15, 135, 168).

[Dei+16] T. Deiß et al. 'Packet forwarding for heterogeneous technologies for integrated fronthaul/backhaul'. In: *2016 European Conference on Networks and Communications (EuCNC)*. June 2016, pages 133–137. DOI: 10.1109/EuCNC.2016.7561019. URL: http://5g-crosshaul.eu/wp-content/uploads/2015/05/5G-Crosshaul-WP3-Latest.pdf (Retrieved: 10th January 2019) (cited on pages 10, 135, 168).

[ECT13] H. E. Egilmez, S. Civanlar and A. M. Tekalp. 'An Optimization Framework for QoS-Enabled Adaptive Video Streaming Over OpenFlow Networks'. In: *IEEE Transactions on Multimedia* 15.3 (April 2013), pages 710–715. ISSN: 1520-9210. DOI: 10.1109/TMM.2012.2232645 (cited on pages 118, 119).

[Fow+14] S. Fowler et al. 'Evaluation and prospects from a measurement campaign on real multimedia traffic in LTE vs. UMTS'. In: *2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace Electronic Systems (VITAE)*. May 2014, pages 1–5. DOI: 10.1109/VITAE.2014.6934475 (cited on page 111).

[GBO14] F. Giust, C. J. Bernardos and A. de la Oliva. 'Analytic Evaluation and Experimental Validation of a Network-Based IPv6 Distributed Mobility Management Solution'. In: *IEEE Transactions on Mobile Computing* 13.11 (November 2014), pages 2484–2497. ISSN: 1536-1233. DOI: 10.1109/TMC.2014.2307304 (cited on pages 69, 70, 76, 78, 79).

[GLB15] F. Giust, **L. Cominardi** and C. J. Bernardos. 'Distributed mobility management for future 5G networks: overview and analysis of existing approaches'. In: *IEEE Communications Magazine* 53.1 (January 2015), pages 142–149. ISSN: 0163-6804. DOI: 10.1109/MCOM.2015.7010527 (cited on pages 15, 96, 168).

[Gro+19] M. Groshev et al. 'A novel approach for multi-access convergence leveraging Edge and Fog computing: an experimentation of a robotics use case'. In: *IEEE Transactions on Services Computing* (2019). To be submitted (cited on page 13).

[Guc+18] J. W. Guck et al. 'Unicast QoS Routing Algorithms for SDN: A Comprehensive Survey and Performance Evaluation'. In: *IEEE Communications Surveys Tutorials* 20.1 (April 2018), pages 388–415. ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2749760 (cited on pages 118, 119).

[Gut+16] P. A. A. Gutiérrez et al. 'NetIDE: All-in-one framework for next generation, composed SDN applications'. In: *2016 IEEE NetSoft Conference and Workshops (NetSoft)*. June 2016, pages 355–356. DOI: 10.1109/NETSOFT.2016.7502408 (cited on page 85).

[HR83] T. Haerder and A. Reuter. 'Principles of Transaction-oriented Database Recovery'. In: *ACM Comput. Surv.* 15.4 (December 1983), pages 287–317. ISSN: 0360-0300. DOI: 10.1145/289.291 (cited on pages 45, 46).

[He+17] T. He et al. 'Location Privacy in Mobile Edge Clouds: A Chaff-Based Approach'. In: *IEEE Journal on Selected Areas in Communications* 35.11 (November 2017), pages 2625–2636. ISSN: 0733-8716. DOI: 10.1109/JSAC.2017.2760179 (cited on page 58).

[HIP15] E. Hernandez-Valencia, S. Izzo and B. Polonsky. 'How will NFV/SDN transform service provider opex?' In: *IEEE Network* 29.3 (May 2015), pages 60–67. ISSN: 0890-8044. DOI: 10.1109/MNET.2015.7113227 (cited on pages 58, 59).

[JM16] L. J. Jagadeesan and V. Mendiratta. 'Programming the Network: Application Software Faults in Software-Defined Networks'. In: *2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. October 2016, pages 125–131. DOI: 10.1109/ISSREW.2016.23 (cited on page 85).

[KLN15] V. A. Kachore, J. Lakshmi and S. K. Nandy. 'Location Obfuscation for Location Data Privacy'. In: *2015 IEEE World Congress on Services*. August 2015, pages 213–220. DOI: 10.1109/SERVICES.2015.39 (cited on page 57).

[KVK14] M. Karimzadeh, L. Valtulina and G. Karagiannis. 'Applying SDN/OpenFlow in Virtualized LTE to support Distributed Mobility Management (DMM)'. In: *Proceedings of the 4th International Conference on Cloud Computing and Services Science (CLOSER 2014)*. SCITEPRESS, April 2014, pages 639–644. ISBN: 978-989-758-019-2. DOI: 10.5220/0004946106390644 (cited on page 90).

[Keh+15] B. Kehoe et al. 'A Survey of Research on Cloud Robotics and Automation'. In: *IEEE Transactions on Automation Science and Engineering* 12.2 (April 2015), pages 398–409. ISSN: 1545-5955. DOI: 10.1109/TASE.2014.2376492 (cited on page 155).

[Kem+12] J. Kempf et al. 'Moving the mobile Evolved Packet Core to the cloud'. In: *2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. October 2012, pages 784–791. DOI: 10.1109/WiMOB.2012.6379165 (cited on page 90).

[Kim+18] J. Kim et al. 'GiLAN Roaming: Roam Like at Home in a Multi-Provider NFV Environment'. In: *IEEE International Symposium on Networks, Computers and Communications (ISNCC)*. June 2018 (cited on pages 16, 163, 169).

[L C+16] **L. Cominardi** et al. '5G-Crosshaul: Towards a unified data-plane for 5G transport networks'. In: *2016 European Conference on Networks and Communications (EuCNC)*. 2016. URL: http://5g-crosshaul.eu/wp-content/uploads/2015/05/WP2-EuCNC16-Reviewers.pdf (Retrieved: 10th January 2019) (cited on pages 10, 135, 168).

[L C+18a] **L. Cominardi** et al. 'Adaptive Telemetry for Software-Defined Mobile Networks'. In: *Journal of Network and Computer Applications* (2018). Submitted (cited on pages 12, 136, 169).

[L C+17a] **L. Cominardi** et al. 'Distributed mobility management solutions for next mobile network architectures'. In: *Computer Networks* 121 (2017), pages 124–136. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2017.04.008. URL: https://e-archivo.uc3m.es/handle/10016/25830 (Retrieved: 10th January 2019) (cited on pages 10, 96, 168).

[L C+18b] **L. Cominardi** et al. 'Experimental evaluation of SDN-based service provisioning in mobile networks'. In: *Computer Standards & Interfaces* 58 (2018), pages 158–166. ISSN: 0920-5489. DOI: 10.1016/j.csi.2018.01.004. URL: https://www.it.uc3m.es/pablo/papers/pdf/2018-cominardi-csi-evaluation.pdf (Retrieved: 10th January 2019) (cited on pages 9, 95, 168).

[L C+18d] **L. Cominardi** et al. 'Opportunities and challenges of Joint Edge and Fog Orchestration'. In: *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. April 2018, pages 344–349. DOI: 10.1109/WCNCW.2018.8369006. URL: https://e-archivo.uc3m.es/bitstream/handle/10016/27036/opportunities_WCNCW_2018_ps.pdf (Retrieved: 10th January 2019) (cited on pages 12, 163, 169).

[L C+18e] **L. Cominardi** et al. 'Understanding QoS Applicability in 5G Transport Networks'. In: *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. June 2018, pages 1–5. DOI: 10.1109/BMSB.2018.8436847. URL: https://e-archivo.uc3m.es/bitstream/handle/10016/27393/understanding_BMSB_2018_ps.pdf (Retrieved: 10th January 2019) (cited on pages 11, 135, 168).

[Lag14] X. Lagrange. 'Very tight coupling between LTE and Wi-Fi for advanced offloading procedures'. In: *2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. April 2014, pages 82–86. DOI: 10.1109/WCNCW.2014.6934865 (cited on page 139).

[LBL12] J. H. Lee, J. M. Bonnin and X. Lagrange. 'Host-based distributed mobility management support protocol for IPv6 mobile networks'. In: *2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* (October 2012), pages 61–68. ISSN: 2160-4886. DOI: 10.1109/WiMOB.2012.6379140 (cited on page 70).

[Li+15] J. Li et al. 'A Hybrid Cloud Approach for Secure Authorized Deduplication'. In: *IEEE Transactions on Parallel and Distributed Systems* 26.5 (May 2015), pages 1206–1216. ISSN: 1045-9219. DOI: 10.1109/TPDS.2014.2318320 (cited on page 141).

[LMR12] L. E. Li, Z. M. Mao and J. Rexford. 'Toward Software-Defined Cellular Networks'. In: *Proceedings of the 2012 European Workshop on Software Defined Networking*. EWSDN '12. IEEE Computer Society, 2012, pages 7–12. ISBN: 978-0-7695-4870-8. DOI: 10.1109/EWSDN.2012.28 (cited on page 69).

[Li+17] X. Li et al. '5G-Crosshaul Network Slicing: Enabling Multi-Tenancy in Mobile Transport Networks'. In: *IEEE Communications Magazine* 55.8 (2017), pages 128–137. ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1600921 (cited on pages 15, 66, 135, 168).

[Li+13] Y. Li et al. 'Software defined networking for distributed mobility management'. In: *2013 IEEE Globecom Workshops (GC Wkshps)*. December 2013, pages 885–889. DOI: 10.1109/GLOCOMW.2013.6825101 (cited on page 90).

[LXX14] Y. Lu, X. Xu and J. Xu. 'Development of a Hybrid Manufacturing Cloud'. In: *Journal of Manufacturing Systems* 33.4 (2014), pages 551–566. ISSN: 0278-6125. DOI: 10.1016/j.jmsy.2014.05.003 (cited on page 141).

[Lyt+16] V. Lytvyn et al. 'A method for constructing recruitment rules based on the analysis of a specialist's competences'. In: *Eastern-European Journal of Enterprise Technologies* 6.2 (December 2016), pages 4–14. DOI: 10.15587/1729-4061.2016.85454 (cited on page 83).

[Mar+18] J. Martín-Pérez et al. 'Modeling MEC deployments for low latency streaming scenarios'. In: *IEEE Transactions on Broadcasting* (2018). Submitted (cited on pages 16, 135, 168).

[May+16] A. Mayoral et al. 'Multi-tenant 5G Network Slicing Architecture with Dynamic Deployment of Virtualized Tenant Management and Orchestration (MANO) Instances'. In: *ECOC 2016; 42nd European Conference on Optical Communication*. September 2016, pages 1–3 (cited on page 59).

[McK+08] N. McKeown et al. 'OpenFlow: Enabling Innovation in Campus Networks'. In: *SIGCOMM Comput. Commun. Rev.* 38.2 (March 2008), pages 69–74. ISSN: 0146-4833. DOI: 10.1145/1355734.1355746 (cited on page 53).

[Men+17] A. Mendiola et al. 'A Survey on the Contributions of Software-Defined Networking to Traffic Engineering'. In: *IEEE Communications Surveys Tutorials* 19.2 (April 2017), pages 918–953. ISSN: 1553-877X. DOI: 10.1109/COMST.2016.2633579 (cited on page 65).

[Met+14] F. Metzger et al. 'Exploratory Analysis of a GGSN's PDP Context Signaling Load'. In: *Journal of Computer Networks and Communications* 2014 (2014). DOI: 10.1155/2014/526231 (cited on page 37).

[Mon+17] R. S. Montero et al. 'Extending the Cloud to the Network Edge'. In: *Computer* 50.4 (April 2017), pages 91–95. ISSN: 0018-9162. DOI: 10.1109/MC.2017.118 (cited on page 139).

[Mos+14] M. Moshref et al. 'Flow-level State Transition As a New Switch Primitive for SDN'. In: *SIGCOMM Comput. Commun. Rev.* 44.4 (August 2014), pages 377–378. ISSN: 0146-4833. DOI: 10.1145/2740070.2631439 (cited on page 122).

[MMM17] E. Münsing, J. Mather and S. Moura. 'Blockchains for decentralized optimization of energy resources in microgrid networks'. In: *2017 IEEE Conference on Control Technology and Applications (CCTA)*. August 2017, pages 2164–2171. DOI: 10.1109/CCTA.2017.8062773 (cited on page 143).

[Nau+] B. Naudts et al. 'How can a mobile service provider reduce costs with software-defined networking?' In: *International Journal of Network Management* 26.1 (), pages 56–72. DOI: 10.1002/nem.1919 (cited on pages 101, 119).

[NBH16] T. T. Nguyen, C. Bonnet and J. Harri. 'SDN-based distributed mobility management for 5G networks'. In: *2016 IEEE Wireless Communications and Networking Conference*. April 2016, pages 1–7. DOI: 10.1109/WCNC.2016.7565106 (cited on pages 60, 89).

[NW13] D. Novak and R. Waterhouse. 'Advanced radio over fiber network technologies'. In: *Opt. Express* 21.19 (September 2013), pages 23001–23006. DOI: 10.1364/OE.21.023001 (cited on page 103).

[Oli+15] A. D. La Oliva et al. 'Xhaul: toward an integrated fronthaul/backhaul architecture in 5G networks'. In: *IEEE Wireless Communications* 22.5 (October 2015), pages 32–40. ISSN: 1536-1284. DOI: 10.1109/MWC.2015.7306535 (cited on page 38).

[Oli+16] A. de la Oliva et al. 'An overview of the CPRI specification and its application to C-RAN-based LTE scenarios'. In: *IEEE Communications Magazine* 54.2 (February 2016), pages 152–159. ISSN: 0163-6804. DOI: 10.1109/MCOM.2016.7402275 (cited on page 100).

[Pan+17] A. Panarello et al. 'Automating the Deployment of Multi-Cloud Applications in Federated Cloud Environments'. In: *Proceedings of the 10th EAI International Conference on Performance Evaluation Methodologies and Tools on 10th EAI International Conference on Performance Evaluation Methodologies and Tools*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2017, pages 194–201. ISBN: 978-1-63190-141-6. DOI: 10.4108/eai.25-10-2016.2266363 (cited on page 141).

[Pen+16] M. Peng et al. 'Fog-computing-based radio access networks: issues and challenges'. In: *IEEE Network* 30.4 (July 2016), pages 46–53. ISSN: 0890-8044. DOI: 10.1109/MNET.2016.7513863 (cited on page 139).

[PWH13] K. Pentikousis, Y. Wang and W. Hu. 'Mobileflow: Toward software-defined mobile networks'. In: *IEEE Communications Magazine* 51.7 (July 2013), pages 44–53. ISSN: 0163-6804 (cited on page 90).

[Rap+18] D. Rapone et al. 'An Integrated, Virtualized Joint Edge and Fog Computing System with Multi-RAT Convergence'. In: *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. June 2018, pages 1–5. DOI: 10.1109/BMSB.2018.8436927 (cited on pages 16, 163, 169).

[Sam+15] M. R. Sama et al. 'Software-defined control of the virtualized mobile packet core'. In: *IEEE Communications Magazine* 53.2 (February 2015), pages 107–115. ISSN: 0163-6804. DOI: 10.1109/MCOM.2015.7045398 (cited on pages 37, 60).

[SCS16] K. Samdanis, X. Costa-Perez and V. Sciancalepore. 'From network sharing to multi-tenancy: The 5G network slice broker'. In: *IEEE Communications Magazine* 54.7 (July 2016), pages 32–39. ISSN: 0163-6804. DOI: 10.1109/MCOM.2016.7514161 (cited on page 39).

[SOM16] M. I. Sanchez, A. de la Oliva and V. Mancuso. 'Experimental Evaluation of an SDN-based Distributed Mobility Management Solution'. In: *Proceedings of the Workshop on Mobility in the Evolving Internet Architecture*. MobiArch '16. New York City, New York: ACM, 2016, pages 31–36. ISBN: 978-1-4503-4257-5. DOI: 10.1145/2980137.2980138 (cited on page 89).

[Sch+16] D. Schulz et al. 'Robust Optical Wireless Link for the Backhaul and Fronthaul of Small Radio Cells'. In: *Journal of Lightwave Technology* 34.6 (March 2016), pages 1523–1532. ISSN: 0733-8724. DOI: 10.1109/JLT.2016.2523801 (cited on page 102).

[Sci+17] V. Sciancalepore et al. 'Mobile traffic forecasting for maximizing 5G network slicing resource utilization'. In: *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. May 2017, pages 1–9. DOI: 10.1109/INFOCOM.2017.8057230 (cited on page 66).

[Sec+15] M. Secondini et al. 'Optical Time–Frequency Packing: Principles, Design, Implementation, and Experimental Demonstration'. In: *Journal of Lightwave Technology* 33.17 (September 2015), pages 3558–3570. ISSN: 0733-8724. DOI: 10.1109/JLT.2015.2443876 (cited on page 103).

[Sim+16] C. Simon et al. '5G exchange for inter-domain resource sharing'. In: *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. June 2016, pages 1–6. DOI: 10.1109/LANMAN.2016.7548842 (cited on pages 140, 144).

[Siv+17] A. Sivanathan et al. 'Characterizing and classifying IoT traffic in smart cities and campuses'. In: *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. May 2017, pages 559–564. DOI: 10.1109/INFCOMW.2017.8116438 (cited on pages 111, 112).

[AOA12] I. Al-Surmi, M. Othman and B. Mohd Ali. 'Mobility management for IP-based next generation mobile networks: Review, challenge and perspective'. In: *Journal of Network and Computer Applications* 35.1 (2012), pages 295–315. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2011.09.001 (cited on pages 67, 69).

[TKF18] T. Taleb, A. Ksentini and P. Frangoudis. 'Follow-Me Cloud: When Cloud Services Follow Mobile Users'. In: *IEEE Transactions on Cloud Computing* (2018), pages 1–1. ISSN: 2168-7161. DOI: 10.1109/TCC.2016.2525987 (cited on page 141).

[TPR14] S. Tomovic, N. Prasad and I. Radusinovic. 'SDN control framework for QoS provisioning'. In: *2014 22nd Telecommunications Forum Telfor (TELFOR)*. November 2014, pages 111–114. DOI: 10.1109/TELFOR.2014.7034369 (cited on pages 118, 119).

[TRP15] S. Tomovic, I. Radusinovic and N. Prasad. 'Performance comparison of QoS routing algorithms applicable to large-scale SDN networks'. In: *IEEE EUROCON 2015 - International Conference on Computer as a Tool (EUROCON)*. September 2015, pages 1–6. DOI: 10.1109/EUROCON.2015.7313698 (cited on pages 118, 119).

[Vai+09] R. Vaishampayan et al. 'Application Driven Comparison of T-MPLS/MPLS-TP and PBB-TE - Driver Choices for Carrier Ethernet'. In: *IEEE INFOCOM Workshops 2009* (2009), pages 1–6. DOI: 10.1109/INFCOMW.2009.5072112 (cited on page 64).

[Val+14] L. Valtulina et al. 'Performance evaluation of a SDN/OpenFlow-based Distributed Mobility Management (DMM) approach in virtualized LTE systems'. In: *2014 IEEE Globecom Workshops (GC Wkshps)*. December 2014, pages 18–23. DOI: 10.1109/GLOCOMW.2014.7063379 (cited on page 91).

[Vil+15] R. Vilalta et al. 'Network virtualization controller for abstraction and control of OpenFlow-enabled multi-tenant multi-technology transport networks'. In: *2015 Optical Fiber Communications Conference and Exhibition (OFC)*. March 2015, pages 1–3. DOI: 10.1364/OFC.2015.Th3J.6 (cited on page 59).

[Wan+15a] D. Wang et al. 'SDN-based joint backhaul and access design for efficient network layer operations'. In: *2015 European Conference on Networks and Communications (EuCNC)*. June 2015, pages 214–218. DOI: 10.1109/EuCNC.2015.7194071 (cited on pages 15, 95, 168).

[Wan+15b] H. Wang et al. 'SoftNet: A software defined decentralized mobile network architecture toward 5G'. In: *IEEE Network* 29.2 (March 2015), pages 16–22. ISSN: 0890-8044 (cited on page 139).

[WB14] Y. Wang and J. Bi. 'A solution for IP mobility support in software defined networks'. In: *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*. August 2014, pages 1–8. DOI: 10.1109/ICCCN.2014.6911783 (cited on page 90).

[Wer+14] M. Wernke et al. 'A classification of location privacy attacks and approaches'. In: *Personal and Ubiquitous Computing* 18.1 (January 2014), pages 163–175. ISSN: 1617-4917. DOI: 10.1007/s00779-012-0633-z (cited on page 57).

[WD14] T. J. Wong and N. Das. 'Modelling and analysis of IEC 61850 for end-to-end delay characteristics with various packet sizes in modern power substation systems'. In: *5th Brunei International Conference on Engineering and Technology (BICET 2014)*. November 2014, pages 1–6. DOI: 10.1049/cp.2014.1073 (cited on pages 111, 112).

[Yap+10a] K.-K. Yap et al. 'Blueprint for Introducing Innovation into Wireless Mobile Networks'. In: *Proceedings of the Second ACM SIGCOMM Workshop on virtualized Infrastructure Systems and Architectures*. VISA '10. New Delhi, India: ACM, 2010, pages 25–32. ISBN: 978-1-4503-0199-2. DOI: 10.1145/1851399.1851404 (cited on page 89).

[Yap+10b] K.-K. Yap et al. 'OpenRoads: Empowering Research in Mobile Networks'. In: *SIGCOMM Comput. Commun. Rev.* 40.1 (January 2010), pages 125–126. ISSN: 0146-4833. DOI: 10.1145/1672308.1672331 (cited on page 90).

[Yap+09] K.-K. Yap et al. 'The Stanford OpenRoads Deployment'. In: WINTECH '09. Beijing, China: ACM, 2009, pages 59–66. ISBN: 978-1-60558-740-0. DOI: 10.1145/1614293.1614304 (cited on page 89).

[YKS14] V. Yazıcı, U. C. Kozat and M. O. Sunay. 'A new control plane for 5G network architecture with a case study on unified handoff, mobility, and routing management'. In: *IEEE Communications Magazine* 52.11 (November 2014), pages 76–85. ISSN: 0163-6804. DOI: 10.1109/MCOM.2014.6957146 (cited on page 60).

[Ye+15] C. Ye et al. 'A First Demonstration of a PON-based Analog Fronthaul Solution Supporting 120 20MHz LTE-A Signals for Future Het-Net Radio Access'. In: *Asia Communications and Photonics Conference 2015*. Optical Society of America, 2015, ASu3E.2. DOI: 10.1364/ACPC.2015.ASu3E.2 (cited on page 103).

Books

[Cou+11] G. Coulouris et al. *Distributed Systems: Concepts and Design*. 5th. Addison-Wesley Publishing Company, 2011. ISBN: 978-0132143011 (cited on page 93).

Patents

[Bla+12] S. Blanc et al. *High Capacity Wireless Communications Systems and Methods*. Patent application US20130294541. E-Blink, May 2012. URL: https://patents.google.com/patent/US20130294541 (Retrieved: 10th January 2019) (cited on page 102).

[LMK18] **L. Cominardi**, A. Mourad and P.-H. Kuo. *Slicing switch resources*. Patent application WO2018106868A1. June 2018. URL: https://patents.google.com/patent/WO2018106868A1/ (Retrieved: 10th January 2019) (cited on pages 16, 135, 168).

[L C+17b] **L. Cominardi** et al. *Methods and apparatus for common transport of backhaul and fronthaul traffic*. Patent application WO2017100394A1. June 2017. URL: https://patents.google.com/patent/WO2017100394A1/ (Retrieved: 10th January 2019) (cited on pages 11, 135, 168).

[L C+17c] **L. Cominardi** et al. *Methods, apparatuses and systems directed to common transport of backhaul and fronthaul traffic*. Patent application WO2017147076A1. August 2017. URL: https://patents.google.com/patent/WO2017147076A1/ (Retrieved: 10th January 2019) (cited on pages 11, 135, 168).

[Per+17a] M. Perras et al. *Mitigating crc calculations in networks that utilize segment routing*. Patent application WO2017172681A1. October 2017. URL: https://patents.google.com/patent/WO2017172681A1/ (Retrieved: 10th January 2019) (cited on page 16).

[Per+17b] M. Perras et al. *Open flow functionality in a software-defined network*. Patent application WO2017142862A1. August 2017. URL: https://patents.google.com/patent/WO2017142862A1/ (Retrieved: 10th January 2019) (cited on pages 11, 136, 169).

Standards

[3GP17] 3GPP. *5G System - Phase 1; CT WG4 Aspects*. Technical Report (TR) 28.891 v15.0.0. 3rd Generation Partnership Project (3GPP), December 2017 (cited on page 67).

[3GP18a] 3GPP. *5G System; Access and Mobility Management Services; Stage 3*. Technical Specification (TS) 29.518 v15.1.0. 3rd Generation Partnership Project (3GPP), September 2018 (cited on page 67).

[3GP18b] 3GPP. *Architecture enhancements for control and user plane separation of EPC nodes*. Technical Specification (TS) 23.214 v15.3.0. 3rd Generation Partnership Project (3GPP), June 2018 (cited on page 60).

[3GP18c] 3GPP. *Evolved General Packet Radio Service (GPRS) Tunnelling Protocol for Control plane (GTPv2-C)*. Technical Specification (TS) 29.274 v15.4.0. 3rd Generation Partnership Project (3GPP), June 2018 (cited on page 74).

[3GP18d] 3GPP. *General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access*. Technical Specification (TS) 23.401 v14.9.0. 3rd Generation Partnership Project (3GPP), September 2018 (cited on page 67).

[3GP18e] 3GPP. *General Packet Radio Service (GPRS); GPRS Tunnelling Protocol (GTP) across the Gn and Gp interface*. Technical Specification (TS) 29.060 v15.2.0. 3rd Generation Partnership Project (3GPP), March 2018 (cited on page 68).

[3GP18f] 3GPP. *General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)*. Technical Specification (TS) 29.281 v15.3.0. 3rd Generation Partnership Project (3GPP), June 2018 (cited on page 73).

[3GP11] 3GPP. *Local IP Access and Selected IP Traffic Offload (LIPA-SIPTO)*. Technical Specification (TS) 23.829 v10.0.1. 3rd Generation Partnership Project (3GPP), October 2011 (cited on page 60).

[3GP18g] 3GPP. *Proxy Mobile IPv6 (PMIPv6) based Mobility and Tunnelling protocols; Stage 3*. Technical Specification (TS) 29.275 v15.0.0. 3rd Generation Partnership Project (3GPP), June 2018 (cited on page 68).

[3GP18h] 3GPP. *S1 Application Protocol (S1AP)*. Technical Specification (TS) 36.413 v15.2.0. 3rd Generation Partnership Project (3GPP), June 2018 (cited on page 72).

[3GP16a] 3GPP. *Service requirements for next generation new services and markets*. Technical Specification (TS) 22.261 v1.0.0. 3rd Generation Partnership Project (3GPP), December 2016 (cited on pages 35, 36).

[3GP18i] 3GPP. *Service requirements for next generation new services and markets*. Technical Specification (TS) 22.261 v16.4.0. 3rd Generation Partnership Project (3GPP), June 2018 (cited on pages 39, 41, 100, 113, 117, 130, 132, 134).

[3GP16b] 3GPP. *Study on Architecture for Next Generation System*. Technical Report (TR) 23.799 v14.0.0. 3rd Generation Partnership Project (3GPP), December 2016 (cited on pages 36, 70, 139).

[3GP14] 3GPP. *Study on enhancements for infrastructure-based data communication between devices*. Technical Specification (TS) 22.807 v13.0.0. 3rd Generation Partnership Project (3GPP), September 2014 (cited on page 63).

[3GP16c] 3GPP. *Study on New Services and Markets Technology Enablers*. Technical Report (TR) 22.891 v14.2.0. 3rd Generation Partnership Project (3GPP), September 2016 (cited on page 38).

[3GP18j] 3GPP. *System Architecture for the 5G System*. Technical Specification (TS) 23.501 v15.3.0. 3rd Generation Partnership Project (3GPP), September 2018 (cited on pages 67, 68, 100, 109, 111).

[BOG18] C. J. Bernardos, A. de la Oliva and F. Giust. *Proxy Mobile IPv6 extensions for Distributed Mobility Management*. Draft draft-ietf-dmm-pmipv6-dlif-03. Internet Engineering Task Force (IETF), October 2018 (cited on pages 69, 70, 86).

[Ber+16] C. J. Bernardos et al. *DETNET crosshauling requirements*. Draft draft-bernardos-detnet-crosshaul-requirements-00. Internet Engineering Task Force (IETF), October 2016. URL: `https://tools.ietf.org/html/draft-bernardos-detnet-crosshaul-requirements-00` (Retrieved: 10th January 2019) (cited on pages 11, 135, 168).

[Bjo10] M. Bjorklund. *YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)*. Request for Comments (RFC) 6020. Internet Engineering Task Force (IETF), October 2010 (cited on page 48).

[Boc+10] M. Bocci et al. *A Framework for MPLS in Transport Networks*. Request for Comments (RFC) 5921. Internet Engineering Task Force (IETF), July 2010 (cited on pages 64, 118).

[Cas+90] J. Case et al. *A Simple Network Management Protocol (SNMP)*. Request for Comments (RFC) 1157. Internet Engineering Task Force (IETF), May 1990 (cited on page 47).

[CP14] H. Chan and K. Pentikousis. *Enhanced mobility anchoring*. Draft draft-chan-dmm-enhanced-mobility-anchoring-00. Internet Engineering Task Force (IETF), July 2014 (cited on page 70).

[Cha+14] H. Chan et al. *Requirements for Distributed Mobility Management*. Request for Comments (RFC) 7333. Internet Engineering Task Force (IETF), August 2014 (cited on page 36).

[Cha+03] K. Chan et al. *Differentiated Services Quality of Service Policy Information Base*. Request for Comments (RFC) 3317. Internet Engineering Task Force (IETF), March 2003 (cited on page 50).

[Che+15] G. Chen et al. *Analysis of Failure Cases in IPv6 Roaming Scenarios*. Request for Comments (RFC) 7445. Internet Engineering Task Force (IETF), March 2015 (cited on page 60).

[Con17] OpenFog Consortium. *OpenFog Reference Architecture for Fog Computing*. OPFRA001.020817. Architecture Working Group, February 2017 (cited on pages 40, 140).

[CPR18] CPRI. *Common Public Radio Interface: eCPRI Interface Specification*. Interface Specification v1.2. Common Public Radio Interface (CPRI), June 2018 (cited on pages 37, 38).

[CPR15] CPRI. *CPRI Specification*. Interface Specification v7.0. Common Public Radio Interface (CPRI), October 2015 (cited on pages 37, 102, 105).

[Dor+10] A. Doria et al. *Forwarding and Control Element Separation (ForCES) Protocol Specification*. Request for Comments (RFC) 5810. Internet Engineering Task Force (IETF), March 2010 (cited on page 44).

[Dro97] R. Droms. *Dynamic Host Configuration Protocol*. Request for Comments (RFC) 2131. Internet Engineering Task Force (IETF), March 1997 (cited on page 64).

[Dro+03] R. Droms et al. *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. Request for Comments (RFC) 3315. Internet Engineering Task Force (IETF), July 2003 (cited on page 73).

[Enn06] R. Enns. *NETCONF Configuration Protocol*. Request for Comments (RFC) 4741. Internet Engineering Task Force (IETF), December 2006 (cited on page 48).

[Enn+11] R. Enns et al. *Network Configuration Protocol (NETCONF)*. Request for Comments (RFC) 6241. Internet Engineering Task Force (IETF), June 2011 (cited on page 48).

[ETS13a] ETSI. *Machine-to-Machine communications (M2M); M2M service requirements*. Technical Specification (TS) 102 689 v2.1.1. European Telecommunications Standards Institute (ETSI), July 2013 (cited on page 63).

[ETS16a] ETSI. *Mobile Edge Computing (MEC); Framework and Reference Architecture*. Group Specification (GS) 003 v1.1.1. European Telecommunications Standards Institute (ETSI), March 2016 (cited on pages 40, 139, 141, 151).

[ETS13b] ETSI. *Network Functions Virtualisation (NFV); Architectural Framework*. Group Specification (GS) NFV 002 v1.1.1. European Telecommunications Standards Institute (ETSI), October 2013 (cited on page 151).

[ETS15] ETSI. *Network Functions Virtualisation (NFV); Ecosystem; Report on SDN Usage in NFV Architectural Framework*. Group Specification (GS) NFV-EVE 005 v1.1.1. European Telecommunications Standards Institute (ETSI), December 2015 (cited on pages 58, 59).

[ETS16b] ETSI. *Network Functions Virtualisation (NFV); Management and Orchestration; Report on Architectural Options*. Group Specification (GS) NFV-IFA 009 v1.1.1. European Telecommunications Standards Institute (ETSI), July 2016 (cited on pages 40, 139, 141).

[ETS18] ETSI. *Network Functions Virtualisation (NFV); Management and Orchestration; Report on Architectural Options to Support Multiple Administrative Domains*. Group Report (GR) NFV-IFA 028 v3.1.1. European Telecommunications Standards Institute (ETSI), January 2018 (cited on page 141).

[ETS14] ETSI. *Requirements for Open Radio equipment Interface (ORI)*. Group Specification (GS) GS ORI 001 V4.1.1. European Telecommunications Standards Institute (ETSI), October 2014 (cited on page 37).

[Gun+08] S. Gundavelli et al. *Proxy Mobile IPv6*. Request for Comments (RFC) 5213. Internet Engineering Task Force (IETF), August 2008 (cited on pages 68, 69, 74).

[Hal+15] E. Haleplidis et al. *Software-Defined Networking (SDN): Layers and Architecture Terminology*. Request for Comments (RFC) 7426. Internet Research Task Force (IRTF), January 2015 (cited on page 62).

[HS10] J. Halpern and J. Hadi Salim. *Forwarding and Control Element Separation (ForCES) Forwarding Element Model*. Request for Comments (RFC) 5812. Internet Engineering Task Force (IETF), March 2010 (cited on page 44).

[IEE17a] IEEE. *10Gb/s Ethernet Passive Optical Network*. Standards for Local and metropolitan area networks 802.11av. Institute of Electrical and Electronics Engineers (IEEE), September 2017 (cited on page 106).

[IEE18a] IEEE. *Bridges and Bridged Networks Amendment: YANG Data Model*. Standards for Local and metropolitan area networks 802.1Qcp. Institute of Electrical and Electronics Engineers (IEEE), May 2018 (cited on page 126).

[IEE09a] IEEE. *Connectivity Fault Management*. Standards for Local and metropolitan area networks 802.1ag. Institute of Electrical and Electronics Engineers (IEEE), November 2009 (cited on pages 117, 118).

[IEE15a] IEEE. *CPRI requirements for Ethernet Fronthaul*. Working Group discussion 802.1CM. Institute of Electrical and Electronics Engineers (IEEE), November 2015. URL: http://www.ieee802.org/1/files/public/docs2015/cm-CPRI-requirements-1115-v01.pdf (Retrieved: 10th January 2019) (cited on page 100).

[IEE18b] IEEE. *Draft Standard for Radio Over Ethernet Encapsulations and Mappings*. NGFI - Next Generation Fronthaul Interface 1914.3. Institute of Electrical and Electronics Engineers (IEEE), June 2018 (cited on pages 104, 108).

[IEE15b] IEEE. *Enhancements for Scheduled Traffic*. Standards for Local and metropolitan area networks 802.1Qbv. Institute of Electrical and Electronics Engineers (IEEE), October 2015 (cited on page 108).

[IEE15c] IEEE. *Frame Preemption*. Standards for Local and metropolitan area networks 802.1Qbu. Institute of Electrical and Electronics Engineers (IEEE), October 2015 (cited on pages 108, 113).

[IEE17b] IEEE. *Frame Replication and Elimination for Reliability*. Standards for Local and metropolitan area networks 802.1CB. Institute of Electrical and Electronics Engineers (IEEE), September 2017 (cited on page 107).

[IEE07] IEEE. *IEEE 802.1D - MAC bridges*. IEEE Standards for Local and metropolitan area networks 802.1D. Institute of Electrical and Electronics Engineers (IEEE), August 2007 (cited on page 63).

[IEE17c] IEEE. *Per-Stream Filtering and Policing*. Standards for Local and metropolitan area networks 802.1Qci. Institute of Electrical and Electronics Engineers (IEEE), September 2017 (cited on page 108).

[IEE16a] IEEE. *Physical Layer and Management Parameters for 25 Gb/s and 40 Gb/s Operation, Types 25GBASE-T and 40GBASE-T*. Standard for Ethernet 802.3bq. Institute of Electrical and Electronics Engineers (IEEE), June 2016 (cited on page 103).

[IEE17d] IEEE. *Physical Layers and Management Parameters for 50 Gb/s, 100 Gb/s, and 200 Gb/s Operation*. Standard for Ethernet 802.3cd. Institute of Electrical and Electronics Engineers (IEEE), December 2017 (cited on page 38).

[IEE09b] IEEE. *Provider Backbone Bridge Traffic Engineering*. IEEE Standards for Local and metropolitan area networks 802.1Q. Institute of Electrical and Electronics Engineers (IEEE), August 2009 (cited on pages 64, 118).

[IEE09c] IEEE. *Provider Backbone Bridges*. Standards for Local and metropolitan area networks 802.1ah. Institute of Electrical and Electronics Engineers (IEEE), March 2009 (cited on pages 45, 108).

[IEE08] IEEE. *Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. PNCS - Precise Networked Clock Synchronization Working Group 1588. Institute of Electrical and Electronics Engineers (IEEE), 2008 (cited on pages 106, 134).

[IEE16b] IEEE. *Standard for Packet-based Fronthaul Transport Networks*. NGFI - Next Generation Fronthaul Interface 1914.1. Institute of Electrical and Electronics Engineers (IEEE), August 2016 (cited on pages 38, 63, 104).

[IEE18c] IEEE. *Standards for Adoption of OpenFog Reference Architecture for Fog Computing*. FOG - Fog Computing and Networking Architecture Framework 1934. Institute of Electrical and Electronics Engineers (IEEE), July 2018 (cited on page 40).

[IEE09d] IEEE. *Station and Media Access Control Connectivity Discovery*. Standards for Local and metropolitan area networks 802.1ab-rev. Institute of Electrical and Electronics Engineers (IEEE), January 2009 (cited on page 63).

[IEE18d] IEEE. *Stream Reservation Protocol (SRP) Enhancements and Performance Improvements*. Standards for Local and metropolitan area networks 802.1Qcc. Institute of Electrical and Electronics Engineers (IEEE), June 2018 (cited on page 108).

[IEE18e] IEEE. *Time-Sensitive Networking for Fronthaul*. Standards for Local and metropolitan area networks 802.1CM. Institute of Electrical and Electronics Engineers (IEEE), March 2018 (cited on pages 104, 108).

[IEE] IEEE. *Time-Sensitive Networking Task Group*. IEEE 802.1 Working Group. Institute of Electrical and Electronics Engineers (IEEE) (cited on pages 104, 107).

[IEE10] IEEE. *Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*. Standards for Local and metropolitan area networks 802.1AS. Institute of Electrical and Electronics Engineers (IEEE), November 2010 (cited on pages 106, 107).

[IEE03] IEEE. *Virtual Bridged Local Area Networks*. IEEE Standards for Local and metropolitan area networks 802.1Q. Institute of Electrical and Electronics Engineers (IEEE), May 2003 (cited on pages 72, 73).

[IEE12] IEEE. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 3: Enhancements for Very High Throughput in the 60 GHz Band*. Wireless LAN Working Group 802.11ad. Institute of Electrical and Electronics Engineers (IEEE), November 2012 (cited on page 106).

[IEE17e] IEEE. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 4: Enhancements for Transit Links Within Bridged Networks*. Wireless LAN Working Group 802.11ak. Institute of Electrical and Electronics Engineers (IEEE), September 2017 (cited on page 108).

[ITU15a] ITU-R. *Framework and overall objectives of the future development of IMT for 2020 and beyond*. M Series Mobile, Radiodetermination, Amateur and Related Satellite Services M.2083. International Telecommunication Union - Radiocommunication Sector (ITU-R), October 2015 (cited on page 38).

[ITU15b] ITU-T. *40-Gigabit-capable passive optical networks (NG-PON2): Definitions, abbreviations and acronyms*. Series G: Transmission Systems and Media, Digital Systems and Networks G.989.1. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), October 2015 (cited on pages 102, 104).

[ITU18a] ITU-T. *Characteristics of Ethernet transport network equipment functional blocks*. Series G: Transmission Systems and Media, Digital Systems and Networks, Series Y: Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks G.8021/Y.1341. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), June 2018 (cited on pages 125, 126).

[ITU18b] ITU-T. *Consideration on 5G transport network reference architecture and bandwidth requirements*. Study Group 15 Contribution 0462. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), February 2018 (cited on pages 101, 112).

[ITU08] ITU-T. *Gigabit-Capable Passive Optical Networks (G-PON): Physical Media Dependent (PMD) Layer Specification*. Series G: Transmission Systems and Media, Digital Systems and Networks G.984.2. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), March 2008 (cited on page 102).

[ITU16a] ITU-T. *Interfaces for the optical transport network*. Series G: Transmission Systems and Media, Digital Systems and Networks G.709. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), June 2016 (cited on page 104).

[ITU16b] ITU-T. *Multichannel bi-directional DWDM applications with port agnostic single-channel optical interfaces*. Draft new Recommendation ITU-T G.metro SG 15 v0.5. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), January 2016 (cited on page 104).

[ITU09] ITU-T. *Multichannel DWDM applications with single-channel optical interfaces*. Series G: Transmission Systems and Media, Digital Systems and Networks G.698.1. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), November 2009 (cited on page 103).

[ITU15c] ITU-T. *Operations, administration and maintenance (OAM) functions and mechanisms for Ethernet based networks*. Series G: Transmission Systems and Media, Digital Systems and Networks, Series Y: Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks G.8013/Y.1731. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), August 2015 (cited on pages 117, 118, 121, 129, 132).

[ITU16c] ITU-T. *Operations, administration and maintenance (OAM) functions and mechanisms for Ethernet based networks*. Series Z: Languages and General Software Aspects for Telecommunication Systems Z.102. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), April 2016 (cited on page 125).

[ITU15d] ITU-T. *Timing characteristics of a synchronous Ethernet equipment slave clock*. Series G: Transmission Systems and Media, Digital Systems and Networks, Series Y: Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks G.8262/Y.1362. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), January 2015 (cited on page 134).

[KA03] H. Khosravi and T. Anderson. *Requirements for Separation of IP Control and Forwarding*. Request for Comments (RFC) 3654. Internet Engineering Task Force (IETF), September 2003 (cited on page 44).

[Lei16] B. Leiba. *Changing the Registration Policy for the NETCONF Capability URNs Registry*. Request for Comments (RFC) 7803. Internet Engineering Task Force (IETF), February 2016 (cited on page 48).

[MM18] J. Medved and D. Meyer. *MPLS-TP Pseudowire Configuration using OpenFlow 1.3*. Draft draft-medved-pwe3-of-config-01. Internet Engineering Task Force (IETF), June 2018 (cited on page 64).

[MEF12a] MEF. *Service OAM Fault Management YANG Module*. Specification MEF 38. Metro Ethernet Forum, April 2012 (cited on page 126).

[MEF12b] MEF. *Service OAM Performance Monitoring YANG Module*. Specification MEF 39. Metro Ethernet Forum, April 2012 (cited on page 126).

[Nar+07] T. Narten et al. *Neighbor Discovery for IP version 6 (IPv6)*. Request for Comments (RFC) 4861. Internet Engineering Task Force (IETF), September 2007 (cited on pages 64, 70, 73, 86).

[ONF14] ONF. *OpenFlow Management and Configuration Protocol: Version 1.2*. Technical Specification (TS) 016. Open Networking Foundation, 2014 (cited on page 62).

[ONF15a] ONF. *OpenFlow switch specification: Version 1.5.1*. Technical Specification (TS) 025. Open Networking Foundation, March 2015 (cited on pages 51, 61, 64, 71, 74).

[ONF15b] ONF. *Relationship of SDN and NFV: Issue 1*. Technical Report (TR) 518. Open Networking Foundation, October 2015 (cited on pages 58, 59).

[ONF16] ONF. *SDN architecture: Issue 1.1*. Technical Report (TR) 521. Open Networking Foundation, January 2016 (cited on pages 59–61, 71, 73).

[Per02] C. Perkins. *IP Mobility Support for IPv4*. Request for Comments (RFC) 3344. Internet Engineering Task Force (IETF), August 2002 (cited on page 74).

[PJA11] C. Perkins, D. Johnson and J. Arkko. *Mobility Support in IPv6*. Request for Comments (RFC) 6275. Internet Engineering Task Force (IETF), July 2011 (cited on pages 69, 74).

[PD13] B. Pfaff and B. Davie. *The Open vSwitch Database Management Protocol*. Request for Comments (RFC) 7047. Internet Engineering Task Force (IETF), December 2013 (cited on page 47).

[Plu82] David C. Plummer. *An Ethernet Address Resolution Protocol*. Request for Comments (RFC) 826. Internet Engineering Task Force (IETF), November 1982 (cited on page 64).

[RLH06] Y. Rekhter, T. Li and S. Hares. *A Border Gateway Protocol 4 (BGP-4)*. Request for Comments (RFC) 4271. Internet Engineering Task Force (IETF), January 2006 (cited on page 48).

[RVC01] E. Rosen, A. Viswanathan and R. Callon. *Multiprotocol Label Switching Architecture*. Request for Comments (RFC) 3031. Internet Engineering Task Force (IETF), January 2001 (cited on page 73).

[Sah+03] R. Sahita et al. *Framework Policy Information Base*. Request for Comments (RFC) 3318. Internet Engineering Task Force (IETF), March 2003 (cited on page 50).

[Sai11] P. Saint-Andre. *Extensible Messaging and Presence Protocol (XMPP): Core*. Request for Comments (RFC) 6120. Internet Engineering Task Force (IETF), March 2011 (cited on page 49).

[SBL14] P. Seite, P. Bertin and JH. Lee. *Dynamic Mobility Anchoring*. Draft draft-seite-dmm-dma-07. Internet Engineering Task Force (IETF), August 2014 (cited on page 70).

[Wan+12] Z. Wang et al. *Analysis of Comparisons between OpenFlow and ForCES*. Draft draft-wang-forces-compare-openflow-forces-01. Internet Engineering Task Force (IETF), March 2012 (cited on page 53).

[Yan+04] L. Yang et al. *Forwarding and Control Element Separation (ForCES) Framework*. Request for Comments (RFC) 3746. Internet Engineering Task Force (IETF), April 2004 (cited on page 44).

[Yeg+18] A. Yegin et al. *On Demand Mobility Management*. Draft draft-ietf-dmm-ondemand-mobility-15. Internet Engineering Task Force (IETF), June 2018 (cited on page 58).

White papers

[5GP14] 5G-PPP. *5G PPP Key Performance Indicators*. 5G Infrastructure Public Private Partnership, 2014. URL: https://5g-ppp.eu/kpis/ (Retrieved: 10th January 2019) (cited on page 100).

[ATT16] AT&T. *ECOMP (Enhanced Control, Orchestration, Management & Policy) Architecture White Paper*. White Paper. AT&T Inc., September 2016 (cited on page 39).

[CIS16] CISCO. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015-2020*. White Paper. June 2016 (cited on page 35).

[Com17] CommScope. *Latency in optical fiber systems*. White Paper. 2017 (cited on page 113).

[Her+16] J.A. Hernández et al. *Detailed analysis of the technologies to be integrated in the XFE based on previous internal reports from WP2/3*. Deliverable 2.1. 5G-Crosshaul consortium, June 2016. URL: http://5g-crosshaul.eu/wp-content/uploads/2018/01/5G-CROSSHAUL_D2.1.pdf (Retrieved: 10th January 2019) (cited on page 9).

[Kek+18] S. Kekki et al. *MEC in 5G networks*. White Paper 28. European Telecommunications Standards Institute (ETSI), June 2018 (cited on page 146).

[L C+18c] **L. Cominardi** et al. *Initial design of 5G-CORAL orchestration and control system*. Deliverable 3.1. 5G-Coral consortium, June 2018. URL: http://5g-coral.eu/wp-content/uploads/2018/06/D3.19802.pdf (Retrieved: 10th January 2019) (cited on page 12).

[Mou+18] A. Mourad et al. *5G-CORAL initial system design, use cases, and requirements*. Deliverable 1.1. 5G-Coral consortium, February 2018. URL: http://5g-coral.eu/wp-content/uploads/2018/04/D1.1_final7760.pdf (Retrieved: 10th January 2019) (cited on page 12).

[Net12] Nokia Siemens Networks. *Signaling is growing 50% faster than data traffic*. White Paper. 2012 (cited on page 37).

[NGM15a] NGMN. *5G White Paper*. White Paper v1.0. Next Generation Mobile Networks Alliance, February 2015 (cited on page 109).

[NGM15b] NGMN. *Backhaul and Fronthaul evolution*. White Paper v1.01. Next Generation Mobile Networks Alliance, March 2015 (cited on page 100).

[NGM16] NGMN. *Description of Network Slicing Concept*. White Paper v1.0. Next Generation Mobile Networks Alliance, February 2016 (cited on page 39).

[ONF12] ONF. *Software-Defined Networking: The New Norm for Networks*. White Paper. Open Networking Foundation, March 2012 (cited on page 35).

[Rez+17] A. Reznik et al. *Developing Software for Multi-Access Edge Computing*. White Paper 20. European Telecommunications Standards Institute (ETSI), September 2017 (cited on pages 16, 163, 169).

[Swa15] T. Swanson. *Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems*. White Paper. R3 CEV, April 2015 (cited on page 143).

List of Acronyms

Symbols

3GPP	3rd Generation Partnership Project
5G-PPP	5G Infrastructure Public Private Partnership

A

AAU	Active Antenna Unit
API	Application Programming Interface
AR	Augmented Reality
ARP	Address Resolution Protocol
ATS	Adaptive Telemetry System

B

BBU	Baseband processing Unit
BGP	Border Gateway Protocol
BSS	Business Support System

C

C-RAN	Centralised RAN
C&M	Control and Managmenet
CCM	Continuity Check Message
CE	Control Element
CFM	Connectivity Fault Management
CMD	Control Mobility Database
CN	Correspondent Node
CoMP	Coordinated Multipoint
CoS	Class of Service
COTS	Commercial Off-The-Shelf
CPRI	Common Public Radio Interface
CWDM	Coarse WDM

D

D2D	Device-to-Device
DHCP	Dynamic Host Configuration Protocol

DMM Distributed Mobility Management
DMM-GW DMM Gateway
DNS Domain Name System
DU Digital Unit
DWDM Dense WDM

E

eCDF empirical Cumulative Distribution Function
eCPRI enhanced CPRI
EFS Edge and Fog computing System
eMBB enhanced Mobile Broadband
eNB Evolved Node B
EPC Evolved Packet Core
EPS Evolved Packet System
ER Egress Router
ETSI European Telecommunications Standards Institute

F

FE Forwarding Element
FLR Frame Loss Ratio
ForCES Forwarding and Control Element Separation
FSM Finite State Machine

G

gNB Next Generation NodeB
GPS Global Positioning System
GTP GPRS Tunnelling Protocol

I

ICMP Internet Control Messagge Protocol
IEEE Institute of Electrical and Electronics Engineers
IETF Internet Engineering Task Force
ILP Integer Linear Programming
IoT Internet of Things
IP Internet Protocol
IQ In-phase and Quadrature
ISG Industry Specification Group
ISP Internet Service Provider
ITU-T ITU Telecommunication Standardization Sector

K

KPI Key Performance Indicator

L

LBM Loopback Message
LBR Loopback Reply
LFB Logical Functional Block
LLC Logical Link Control
LLDP Link Layer Discovery Protocol
LMA Local Mobility Anchor
LTE Long Term Evolution

LTM Linktrace Message
LTR Linktrace Reply

M

MAG Mobile Access Gateway
MANO Management and Orchestration
MEC Mobile Edge Computing
MEF Metro Ethernet Forum
MIB Management Information Base
MIMO Multiple-Input and Multiple-Output
MIoT Massive Internet of Things
MIPv6 Mobile IPv6
MM Mobility Management
MME Mobility Management Entity
MN Mobile Node
MPC Mobile Packet Core
MPLS Multiprotocol Label Switching
MPLS-TP MPLS Transport Profile
MTC Machine Type Communication

N

NAT Network Address Translation
NC Network Controller
NE Network Element
NETCONF Network Configuration Protocol
NFV Network Function Virtualisation
NFVI NFV Infrastructure
NGFI Next Generation Fronthaul Interface
NGMN Next Generation Mobile Networks
NLoS non-line-of-sight

O

OAM Operations, Administration and Maintenance
OCS Orchestration and Control System
OID Object Identifier
ONF Open Networking Foundation
ORI Open Radio equipment Interface
OSS Operation Support System
OTN Optical Transport Network

P

PBA Proxy Binding Acknowledgment
PBB Provider Backbone Bridges
PBB-TE PBB Traffic Engineering
PBU Proxy Binding Update
PCE Path Computation Element
PGW Packet Data Network Gateway
PMIPv6 Proxy Mobile IPv6
PoC Proof of Concept
PON Passive Optical Network
PoP Point of Presence

PtMP point-to-multipoint
PtP point-to-point

Q

QoE Quality of Exprerience
QoS Quality of Service

R

RA Router Advertisement
RAN Radio Access Network
RAT Radio Access Technology
REC RE Control
REC Radio Equipment
RoE Radio over Ethernet
RoF Radio over Fiber
RPC Remote Procedure Call
RRH Remote Radio Head
RS Router Solicitation
RTT Round Trip Time
RU Radio Unit

S

SBI Southbound Interface
SDH Synchronous Digital Hierarchy
SDL Specification and Description Language
SDN Software Defined Networking
SGW Serving Gateway
SLA Service Level Agreement
SNAP Subnetwork Access Protocol
SNMP Simple Network Management Protocol

T

TDM Time-division Multiplexing
TEID Tunnel End-Point Identifier
TFT Traffic Flow Template
TLS Transport Layer Security
TTL Time to Live
TWDM Time and Wavelength Division Multiplexing

U

UE User Equipment
UPF User Plane Function
URI Uniform Resource Identifier
URLLC Ultra-Reliable and Low Latency Communications

V

V-RAN Virtualised RAN
V2I Vehicle-to-Infrastructure
VIM Virtualised Infrastructure Manager
VLAN Virtual LAN
VM Virtual Machine

VNF Virtual Network Function
VPN Virtual Private Network

W

WDM Wavelength Division Multiplexing
WG Working Group

X

XCF Crosshaul Common Frame
XCSE Crosshaul Circuit Switching Element
XFE Crosshaul Forwarding Element
XML Extensible Markup Language
XMPP Extensible Messaging and Presence Protocol
XPFE Crosshaul Packet Forwarding Element