Applying backpressure to balance resource usage in software-defined wireless backhauls

Jorge Baranda José Núñez-Martínez Josep Mangues-Bafalluy

Centre Tecnològic de Telecomunicacions de Catalunya (CTTC)
Av. Carl Friedrich Gauss, 7
08860 Castelldefels (Barcelona), Spain
e-mail: [jorge.baranda, jose. nunez, josep.mangues]@cttc.cat

***Abstract*—The expected higher complexity of upcoming 5G wireless backhauls suggests the need to evolve towards a software-defined networking (SDN) paradigm to increase the degree of programmability of these networks. Within the context of software-defined wireless backhaul, one important issue is the provision of an even resource consumption of both network and IT resources. Backpressure policies have shown their ability to balance resource consumption in traditional (non-SDN) wireless backhauls. This paper analyzes two use cases in which backpressure policies can be integrated in software-defined wireless backhaul to manage network and IT resources. The first use case proposes an SDN application based on a centralized backpressure policy to balance network resources in the wireless backhaul. Simulation results reveal how the granularity of routing decisions in the SDN application significantly affects the data plane performance, suggesting a trade-off between data plane performance and overload in the control plane. The second use case proposes a distributed backpressure policy to deal with the management of computing resources by balancing the processing load caused by OpenFlow (OF) switch requests among the available distributed SDN controllers. Simulation results demonstrate how a dynamic and distributed backpressure policy can balance the processing load amongst different SDN controllers, hence, leading to significant improvements with respect to static mapping policies.**

I. Introduction

The management of wireless backhaul resources is expected to increase in complexity even more for future 5G mobile networks. This trend can be explained, between others, due to the following factors. On one hand, the expected higher density of low power base stations, referred to as small cells (SC), which is an effective way to increase network capacity due to reduced cell radii. On the other hand, the higher network dynamicity due to user mobility, decrease of per-device reliability, abrupt changing traffic loads, and mixed Quality of Service (QoS) requirements. Thus, given the tight requirements of SC backhauls [1] and being unlikely that fiber reaches every SC site in an ultra-dense deployment, the management of wireless, and possibly wireless mesh, backhaul resources formed by SCs will have to be significantly improved.

As in [2], we posit that the management of wireless backhaul resources can be improved by applying the principles of Software-Defined Networking (SDN), forming a software-defined wireless backhaul (see Fig. 1). SDN [3] is a game-changing paradigm allowing the idea of "programmable networks", separating the control plane from network devices. In SDN, the control plane is moved to a logically centralized software element referred to as the SDN controller. By means of the northbound interface (NBI), developed SDN applications direct specific functions through the SDN controller, which interacts with the physical infrastructure, that is, the network devices. SCs embedding backhaul network devices form the data plane, which is programmed by means of an open interface, the so-called southbound interface (SBI). The de-facto protocol for this interface is the OpenFlow (OF) protocol [4], which has a strong support from industry, research, and academia. OpenFlow is developed by the Open Networking Foundation (ONF) [5], which is dedicated to accelerating the adoption of open SDN networks.

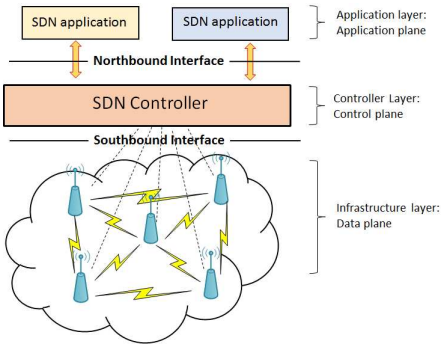


Fig. 1: Software-defined wireless backhaul

In the context of a software-defined wireless backhaul, a key question is how to provide a proper load balancing of network and IT resources. Regarding the load balancing of network resources, several solutions in SDN deployments have been proposed in the context of data centers [6], [7]. In the case of load balancing of IT resources, ElastiCon [8] proposes a centralized algorithm to change the control assignation of switches among the pool of available SDN controllers. Thus, the processing load generated by their requests is distributed evenly among the resources provided by the SDN controllers. However, literature lacks from such efforts for emerging software-defined wireless backhauls.

On the other hand, backpressure [9] has been shown as an effective approach to balance traffic. Backpressure algorithms are based on minimizing the queue backlog differentials between neighboring nodes to push network resources towards lower congestion states. For this reason, backpressure routing is able to exploit several paths between source and destination

SCs. Practical extensions of backpressure has been developed in legacy distributed wireless multi-hop backhauls [10], [11].

The contribution of this paper is the analysis of two research problems for software-defined wireless backhauls where we foresee that a backpressure approach could be adopted for a proper load balancing of resources. To our knowledge, this is the first study integrating backpressure-based policies for the management of software-defined wireless backhauls. In the first use case, we leverage a backpressure policy to balance network resources. In particular, we propose an SDN application based on a centralized backpressure approach that balances traffic flows in a software-defined wireless backhaul. The goal here is to distribute traffic flows to improve the usage of network resources. Different from our previous work [10], [11], where routing decisions were taken on a per-packet granularity, the centralized backpressure policy performs routing decisions with coarser granularity, that is, per-flow statically and on a periodical basis. In the second use case, we propose a distributed backpressure algorithm to balance the processing load among available SDN controllers due to uneven generation of requests by OF switches forming the software-defined wireless backhaul. In contrast to the centralized algorithm in [8], we propose a distributed backpressure approach embedded in the SDN controllers to dynamically map the control of OF switches across the multiple SDN controller instances. The goal here is to balance the number of OF switch requests served at each instance of the SDN controllers. Note that in this case, the resource balanced is not network but IT resources. In addition, we suggest the changes to be introduced in a real SDN deployment to add the functionalities described by both use cases. OpenFlow v1.3 [4] is used as a baseline because currently, this is the most updated version of the OF protocol that supports most of the current implementation of OF switches. Furthermore, we provide initial simulation results with ns-3 [12] and Matlab that indicate the convenience and potential of using backpressure policies due to their simplicity and efficiency.

The rest of this paper is organized as follows. In Section II, we provide a review of the related work. Section III focuses on the design, and simulation of the centralized SDN application based on backpressure for load balancing of traffic flows. Section IV provides a description and study of the backpressure approach to balance the mapping of SCs embedding OF switches to SDN controllers. The paper concludes with Section V.

II. Related Work

In this section, we first summarize the main ideas behind the backpressure algorithm applied to legacy distributed wireless networks. Next, we detail related work on the two research topics where we foresee that a backpressure policy can apply.

A. Backpressure Concept

The origins of the Backpressure concept lies on the seminal paper of Tassiulas and Ephremides [9]. In essence, it is a centralized policy to route traffic in a multi-hop network which attains throughput optimality by minimizing the Lyapunov drift in the network. That is, this policy aims at minimizing the sum of the queue backlogs in the network from one time slot to the following one. This is achieved by transmitting packets at each time slot between network elements so the queue backlog differentials are minimized. The backpressure algorithm was afterwards extended by Neely and Modiano [13], proposing to combine the previous scheme with a drift-plus-penalty technique to optimize the performance of wireless multi-hop networks. The theoretical strengths derived from this work have recently increased the interest on practical implementations in the context of wireless multi-hop backhauls [10], [11]. Experimental and simulation results showed that the resulting distributed backpressure routing strategy attains an even network resource consumption in a non-SDN wireless backhaul, where each SC participates in the distributed control plane. In contrast, this paper pursues the application of backpressure-based policies in software-defined wireless backhauls.

B. SDN Research Topics where Backpressure can apply

1) Load Balancer of traffic flows as SDN Application: The SDN paradigm eases the creation and deployment of more complex network applications, such as the reduction of power consumption [14]. As pointed out by the ONF [5], efficient load balancing solutions are needed in highly dynamic networks such as the ones managed based on the SDN paradigm. In the context of software-defined wireless backhaul networks, an SDN application performing efficient load balancing of network resources gains more importance due to the limited nature of such resources that a wireless backhaul has compared to its wired counterpart. Regarding the use of load balancing SDN applications, most of the work is derived from the Equal Cost Multipath (ECMP) protocol [15] and is mostly conceived for the field of data centers [6], [7]. In this paper, we propose a centralized approach to choose paths for incoming flows using a backpressure policy, that is, merely based on polling queue backlog information of switch ports. Unlike in [7], which performs the path selection measuring the bandwidth consumed by each flow at each constrained link. Thus, we provide another alternative to the static resource allocation solution (shortest-path routing), which does not exploit the path redundancy available in the backhaul deployment.

2) Balancing the processing load across multiple distributed SDN controllers: A distributed architecture for the SDN controller is justified by several reasons: (1) administrative issues, (2) scalability, (3) fault-tolerance, and (4) switch to controller latency reduction. The implementation of distributed systems entails well-known challenges, as pointed out in the SDN architecture document of the ONF [5]. Furthermore, the load distribution (i.e., number of managed OF switches) among the available SDN controllers constitutes an open problem in the context of a distributed (but still logically centralized) SDN controller architecture. As mentioned in [8], a static mapping of OF switches to SDN controllers can produce uneven load distributions in the SDN controllers derived from changes in the traffic characteristics. Consequently, some controllers experience overloading, causing an increase of network latency associated to the request response time. Hence, a distributed deployment of SDN controllers requires of procedures and algorithms to dynamically assign switches to controllers. In this way, the switches (and its generated load) can be balanced among the available resources at the pool of SDN controllers. In [8], an example of such kind of procedure run by a central entity is presented. In this paper, we present a distributed

backpressure algorithm to dynamically assign switches to SDN controllers. Besides, and in contrast to [8], our algorithm considers the distance from switches to controllers as a parameter to constraint the mapping of switches to controllers.

III. SDN Application: Load Balancer of traffic flows

This section details the application of backpressure as a centralized SDN application to achieve an efficient balancing of traffic flows across multiple OF switches embedded in SC devices, as showed in Fig. 1. As explained in [16], an SDN controller contains, among others, two basic modules: the link discovery and the topology manager. The link discovery module discovers the nodes and maintains the information about the status of the physical links in the network. The topology manager builds and maintains the topology information and calculates the routes in the network using the information collected by the link discovery module.

When a OF switch receives a packet, it looks for a matching forwarding rule in its flow table. If there is a match the packet is forwarded. If there is not a match, the packet is encapsulated in a OF *OFPT_PACKET_IN* request towards the SDN controller, which is in charge of providing the rule to forward this packet. To compute routes, the topology manager embedded in the SDN controller leverages the Dijkstra's algorithm to find the shortest path between two network elements, deriving in a static allocation of network resources. However, this approach does not ensure an efficient use of the network resources depending on the network congestion conditions. We propose to enhance the routing functionality with an SDN application that uses a backpressure policy to provide routing and load balancing for the traffic traversing the software-defined wireless backhaul. In order to do so, the application bases on the information provided by the SDN controller services such as the topology manager and the link discovery.

The SDN application needs to maintain an abstracted global view of the network to determine the appropriate route for an incoming flow and to install the corresponding rules in the OF switches. The information about the network topology and link status is gathered from the database maintained by the link discovery and the topology manager modules. The information in this database is populated running a discovery procedure using the Link Layer Discovery Protocol (LLDP). With this information, the SDN application can obtain the multiple routes between any pair of network elements embedding a protocol, such as Equal Cost Multi-Path (ECMP) [15]. To use a backpressure policy, the SDN application also requires to poll periodically the queue sizes of the ports in the OF switches embedded in the SCs.

In particular, the SDN application assigns a weight w to each possible link (i, j) with rate R_{ij} forming part of the routes available for this traffic flow according to the following expression:

$$w_{ij} = (Q_i - Q_j)R_{ij}, \tag{1}$$

where Q_i is the queue occupancy of OF switch i and Q_j is the queue occupancy of OF switch j. For each new flow entering the network, weights are calculated out of the available routes for this flow. Once all the weights are computed, we select the path p^* satisfying:

$$p^* = \arg\max_{p \in P} \sum_{(i,j) \in p} w_{ij} \tag{2}$$

where p is one of the possible end-to-end paths of the set P calculated by the SDN application. In this way, the application selects, for each flow, the route that maximizes the path weight computation depicted in Equation (2). Note that the weights can be periodically recalculated configuring appropriately the timeouts of the installed flow table rules at the OF switches. This is of special interest in the case of long-lived flows since it allows rerouting, hence experiencing path readjustments to better adapt to abrupt changes in the network congestion conditions. Additionally, path readjustments may be necessary due to the changing conditions of the wireless medium.

In order to implement this load balancing application, some adjustments to the OF v1.3 specification [4] are required. The application needs to gather the queue sizes to abstract the congestion level of ports in the OF switches forming the backhaul network. Periodically, the OF switches can send this information to the SDN controller to be stored in the network topology database. We can leverage OF multipart messages, which are primarily used to request statistics or state information from the switch.

In particular, the *OFPMP_QUEUE* multipart message provides queue statistics. However, the current implementation of this message does not include any field related to the queue size of a port in packets. We propose to expand this message by including queue occupancy information to infer network congestion conditions. The link rate information between OF switches is available by means of the asynchronous *OFPT_PORT_STATUS* message. OF switches send this message to inform the controller of changes on a port. The port physical rate is encoded in the *OFP_PORT* structure, which describes the information associated to a port.

Thus, for each OF *OFPT_PACKET_IN* request received by the SDN controller, instead of determining the route running a static single shortest path protocol, it will request to the SDN application the appropriate route. The queue and routing information exchanged between the application and the controller is collected through the NBI by means of the REST API using the HTTP protocol. From those available routes, the SDN application selects the route maximizing the sum of the link weights computation, and informs the SDN controller of the selected route. Through the OF *OFPT_FLOW_MOD* message, the controller will then push the new flow rule to the network switches.

A. Case Study

In this section, we evaluate the modeled centralized SDN application based on backpressure with ns-3 [12]. In particular, *BP(per-flow)* takes routing and load balancing decisions on a per-flow basis, and *BP(periodical)* on a periodic basis, thus, potentially changing the path for long-lived flows. We compare these approaches with our backpressure distributed solution in [11], labeled as *BP(per-packet)*, which takes decisions on a per-packet basis in a non-SDN version of the evaluated scenario, and a centralized shortest-path based route determination. For these initial results, we assume that the control plane is part of another alternative reliable network so that control and data plane do not share the same infrastructure.

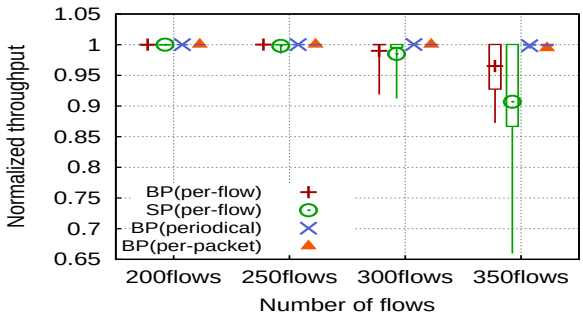


Fig. 2: Performance comparison of achieved throughput

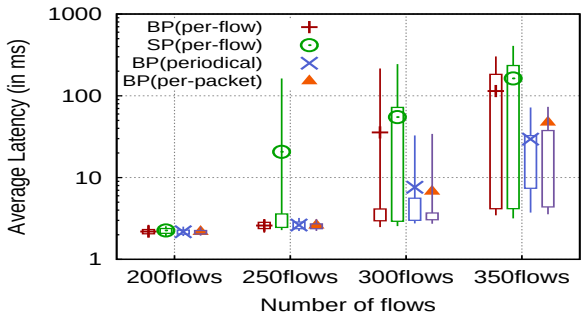


Fig. 3: Performance comparison of achieved latency

Fig. 2 and Fig. 3 shows the statistical distribution of the averaged performance of 25 repetitions of the simulated 5x5 grid like wireless mesh backhaul with an SDN controller. Simulations span over five minutes with multiple constant bit rate (CBR) traffic flows entering and leaving the network, thus sharing the wireless resources. We show the normalized throughput and the average latency in the data plane for a different number of input flows. The normalized throughput is defined by the ratio between the received and the injected traffic. For each of these network performance metrics, we use average values and boxplots to represent their statistical distribution. Boxplots represent the statistical distribution stretching from the 25th to the 75th percentiles, and the whiskers represent the 5th and 95th percentiles.

The trend observed in Fig. 2 and Fig. 3 shows that the obtained gains with *BP(per-flow)* and *BP(periodical)* are achieved when network dynamics increase because the available resources in the data plane of the wireless backhaul are used more efficiently. Remarkably *BP(per-flow)* and *BP(periodical)*, obtain a minimum improvement of 40% on average latency compared to *SP(per-flow)* for the case of 350 flows. In general, *BP(periodical)* tends to show better performance in terms of normalized throughput and latency than *BP(per-flow)* and *SP(per-flow)* for high loads. The periodical recomputation of a path for a flow increases the degree of traffic distribution showing less variability and significant improvement in terms of throughput and latency compared to *SP(per-flow)* and *BP(per-flow)*.

Although the performance in terms of latency of *BP(periodical)* is even better than that of *BP(per-packet)* for the 350 flows case, in general we observe similar behavior with *BP(periodical)* and *BP(per-packet)*. This suggests that *BP(periodical)* could offer a good trade-off between data plane performance and overhead (e.g., Packet In requests) generated by OF control plane traffic. In our simulations, the path recomputation was performed every second. The impact of the periodicity of such recomputation will be a subject of further study. Note that control plane overhead acquires even more relevance in scenarios where control and data plane share resources in an unreliable wireless backhaul network.

IV. Balancing the load between multiple distributed SDN controllers

In the envisioned context of high network dinamicity, the rate of requests generated in the set of OF switches controlled by an SDN controller experience variations due to changes in the spatial traffic distribution. The processing load at an SDN controller is directly related with the amount of requests generated by the OF switches embedded in SCs. As explained in [8], a dynamic assignment of switches to controllers could react in front of processing overloads at the SDN controllers in such kind of distributed deployments. Due to this, in this section, we focus on the modeling of a distributed backpressure algorithm that pursues an even distribution of the processing load amongst the pool of available SDN controllers present in the network deployment.

The proposed algorithm considers that the software-defined wireless backhaul has been divided into different controller domains to keep bounded the latency in the communication between switches and controller as depicted in Fig. 4. Each different domain contains an SDN controller, which initially manages a set of OF switches deployed in the given domain. Each SDN controller embeds a module which records the number of OF requests from each switch under control during the last observation time. This value abstracts the queue backlog of requests associated to each SDN controller. Moreover, it is disseminated periodically to adjacent SDN controllers through the east/westbound interfaces, which allow inter-SDN controller communications. We consider that switches can only be migrated between adjacent SDN controller domains to avoid excessive latencies in the communication between the OF switch and SDN controller. In addition, we assume that only one switch migration is allowed between a pair of SDN controllers at each scheduling decision period (time slot) to decrease the impact of disruption on ongoing flows due to the migration process.

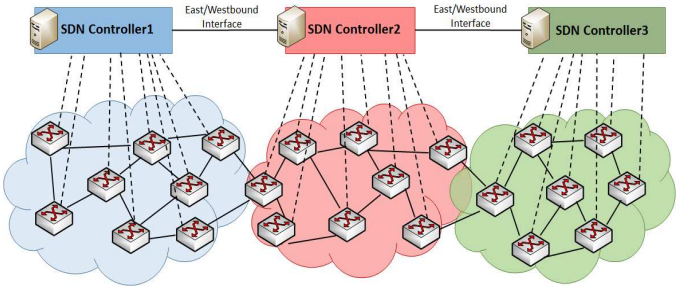


Fig. 4: Distributed SDN controller deployment

At the beginning of each time slot, each SDN controller checks if its processing load during the last time slot has exceeded a predefined threshold denoted by U_t. This threshold pretends to avoid continuous migrations of OF switches between SDN controllers and is fixed assuming that the SDN controller is able to serve the incoming requests in a reasonable

time. If U_t is exceeded, this means that the SDN controller is overloaded and needs to migrate OF switches to neighboring SDN controllers to reduce/balance its processing load. The SDN controller "under stress" computes the following metric, which seeks the minimization of the Lyapunov drift between the processing load of the local SDN controller and the neighboring SDN controllers:

$$s_{ij} = \max(0, \frac{(R_i - R_j)}{2}) \quad (3)$$

where s_{ij} denotes the number of requests which should be migrated to attain balance between the considered SDN controllers. R_i is the number of requests in the local SDN controller and R_j denotes the reported number of requests of the neighboring SDN controller.

Each SDN controller keeps track of the neighboring SDN controllers j with positive weight s_{ij}. Remember that we assume that OF switches can only be migrated between adjacent domains. Thus, according to Fig. 4, a switch originally associated to the blue domain can be part of the red domain but not a candidate to be controlled by the green domain.

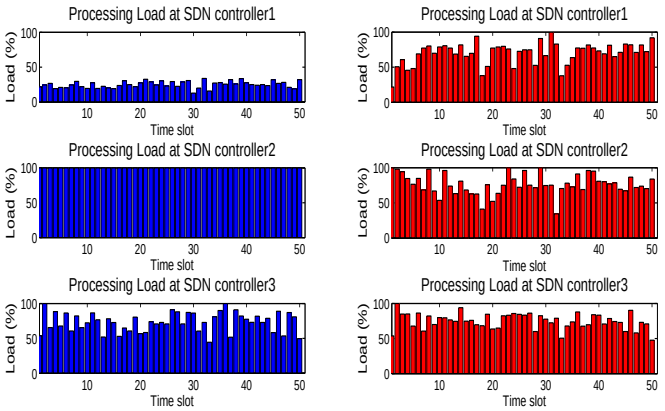
The switch selection algorithm is executed as follows. First, neighboring SDN controllers j with positive weight s_{ij} are sorted in descending order. Thus, those neighboring SDN controllers presenting a lower processing load are considered before to host a switch migration. Second, we generate the list of local candidate switches to be migrated. This list is populated according to the spatial location of an OF switch. Third, we select the switch from this list that produces the biggest reduction in s_{ij} while not overloading the neighboring SDN controller j, that is, not exceeding the previous defined utilization threshold U_t. In this way, we are applying a max-weight (or backpressure) policy to select intended OF switch migrations. Then, the algorithm repeats the process for the following neighboring SDN controllers j of the list. Prior to this, the weight s_{ij} is recomputed in the local SDN controller. The reason to update the weights is to avoid unnecessary OF switch migrations. Once the local SDN controller has finished the previous process, it notifies the corresponding neighboring SDN controllers and the migration operations start if any switch has been selected. Note that with the proposed assumptions, OF switch migrations per time slot for an SDN controller are bounded to the the number of neighboring SDN controllers. Thus, the process to decrease the overload of an SDN controller may imply several time slots.

Since version 1.3 [4], OF protocol defines three operational modes for a controller: master, slave, and equal. This allows the design of switch migration protocols between SDN controllers, such as the one presented in [8], which ensures minimal disruption to ongoing flows. In this protocol, the original controller changes its role to slave by sending the corresponding OF *OFPT_ROLE_REQUEST* message to the switch. The new SDN master controller must send the OF *OFPT_ROLE_REQUEST* message to inform the switch of its new role as master. The other requirement to implement our distributed algorithm is the necessity of data exchange between SDN controllers through the east/westbound interfaces. An example of protocol to manage east/westbound communications between neighboring SDN controllers is the Advanced Message Queuing Protocol (AMQP) [17], as proposed in [18]. AMQP is a lightweight protocol that allows the exchange of information between entities, hence being a suitable alternative to exchange the information related to the number of received requests in each SDN controller. Note, however, that currently, there is not a clear east/westbound interface standard, like OF for the SBI, to provide compatibility and interoperability between different SDN controllers.

A. Case Study

In this subsection, we provide simulation results obtained with Matlab software where we compare the proposed load balancing algorithm based on backpressure with a static switch-controller mapping policy. We consider three SDN controllers deployed as showed in Fig. 4, where each controller initially manages a disjoint set of ten OF switches. Each set of switches models the spatial distribution of OF requests in the network. We assume that each SDN controller can manage up to 10000 requests per time slot. Each switch generates independently OF requests in each slot following a uniform random distribution. Switches of set one generate OF requests uniformly distributed in the range [0,500], switches of set two in the range [0,2500], and switches of set three in the range [0,1500]. Hence, initially, each SDN controller receives a uniform distribution of OF requests at a mean rate of 2500, 12500, and 7500 requests per time slot, respectively.

Fig. 5 shows the evolution of the processing load at each controller comparing the different switch-controller mapping strategies. Fig. 5(a) reveals that SDN controller2 suffers from processing overload as a consequence of the static mapping strategy, which derives in an increase of the response time for the received requests. However, the other SDN controllers have spare processing resources which could be used to reduce the processing load at SDN controller2. In contrast, when using the proposed algorithm based on backpressure, the processing load of SDN controller2 is mostly delegated to SDN controller1 attaining a fair share of load between the available controllers, as can be observed in Fig. 5(b).



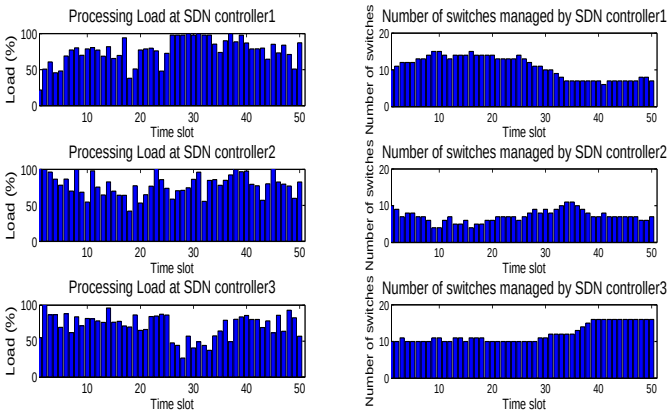
(a) Static mapping (b) Dynamic mapping

Fig. 5: Evolution of processing load at each SDN controller

Next, we show the reaction of the proposed algorithm when switches of each set change suddenly the mean number of generated requests, simulating a spatial variation in network traffic conditions. In particular, the mean rate of requests of

each group of switches is changed from 2500, 12500, and 7500 requests to 12500, 7500 and 2500 requests, respectively, in time slot 25. Fig. 6(a) shows the evolution of the processing load experienced by each SDN controller, and Fig. 6(b) shows the temporal evolution of the number of OF switches managed by each SDN controller. Initially, we observe how the overload suffered by SDN controller2 is delegated mostly to SDN controller1. At time slot 25, the abrupt change in the request generation rate produces overload in SDN controller1. As a result, this controller starts assigning switches to SDN controller2. Fig. 6(b) shows that this process implies several time slots due to the design constraints of only migrating one switch per neighbor and time slot. Then, SDN controller2 starts experiencing an elevated processing load and ends up transferring switches to SDN controller3. Finally, the system becomes stable and switch migrations are less frequent.

The above results suggest that workload amongst SDN controllers can be shared using a distributed backpressure policy, confirming its suitability to balance IT resource consumption (i.e., load in SDN controller). Initial simulation results, in general, reveal a good reaction of this policy even for abrupt changes in the workload to be managed by an SDN controller.



(a) Processing load (b) Managed switches

Fig. 6: Dynamic mapping when changing traffic conditions

V. Conclusions

In this paper, we identify two use cases relevant for a software-defined wireless backhaul where backpressure based policies can be applied: load balancing of traffic flows and balancing of processing load among the available SDN controllers. As for the former, we devise an SDN application based on a centralized backpressure policy. We also provide hints on how to implement such an application in a real OF context. Simulation conducted for this use case demonstrate that a centralized backpressure policy with proper periodic route recomputations can bring significant performance gains for the data plane. As for the latter, we propose and provide some initial simulation results of a distributed backpressure approach to balance processing load amongst a physically distributed SDN controller architecture. We also provide some hints on how to implement such a distributed dissemination of load information between neighboring SDN controllers.

To the best of our knowledge, this paper is the first one focusing on software-defined wireless backhauls and backpressure policies to manage not only network resources but also IT resources. We expect that it constitutes a starting point for the introduction of such technique and, more in general, the introduction of stochastic network optimization techniques to better manage the available resources in software-defined wireless backhauls. The efficient management of resources in such kind of deployments is a vital issue due to their associated constraints compared to their wired counterparts.

Acknowledgment

This work was carried out in part within H2020 SANSA project, funded by EC under grant agreement no. 645047, and by the Spanish Ministry of Economy and Competitiveness under grant TEC2014-60491-R.

References

[1] Next Generation Mobile Networks Alliance, "Small cell backhaul requirements," *NGMN White paper*, 2012.

[2] C.J. Bernardos, A. de la Oliva, P. Serrano, A. Banchs, L.M. Contreras, Hao Jin, and J.C. Zuñiga, "An architecture for software defined wireless networking," *Wireless Communications, IEEE*, vol. 21, no. 3, pp. 52–61, June 2014.

[3] N. McKeown, "Software-Defined Networking," INFOCOM Keynote talk, April 2009.

[4] Open Networking Foundation, "OpenFlow Switch Specification (Version 1.3.0)," June 2012.

[5] Open Networking Foundation, "Available at: https://www.opennetworking.org," .

[6] C.A.B. Macapuna, C.E. Rothenberg, and M.F. Magalha?es, "In-packet bloom filter based data center networking with distributed openflow controllers," in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, Dec 2010, pp. 584–588.

[7] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proc. of the 7th USENIX Conf. on Networked Systems Design and Implementation*, Berkeley, CA, USA, 2010, NSDI'10, pp. 19–19, USENIX Association.

[8] Advait D., F. Hao, S. Mujherjee, T.V. Lakshman, and R. Kompella, "Towards an elastic distributed SDN controller," in *Proc. of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN 2013)*. ACM, 2013, pp. 7–12.

[9] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, Nov. 1992.

[10] J. Núñez-Martínez, J. Baranda, and J. Mangues-Bafalluy, "Experimental evaluation of self-organized backpressure routing in a wireless mesh backhaul of small cells," *Ad Hoc Networks, Elsevier*, 2015.

[11] J. Núñez-Martínez, J. Baranda, and J. Mangues-Bafalluy, "A self-organized backpressure routing scheme for dynamic small cell deployments," *Ad Hoc Networks, Elsevier*, 2015.

[12] "The ns-3 network simulator, Avalaible at: http://www.nsam.org," .

[13] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 1, pp. 89–103, 2005.

[14] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastictree: Saving energy in data center networks," in *Procs of the 7th USENIX Conf on Networked Systems Design and Implementation*, Berkeley, CA, USA, 2010, NSDI'10, pp. 17–17, USENIX Association.

[15] C. Hopps, "Analysis of an equal-cost multi-path algorithm," 2000.

[16] Aricent, "Demystifying routing services in software-defined networking," *White Paper*, 2013.

[17] Advanced Message Queuing Protocol, "Available at: http://www.amqp.org," .

[18] K. Phemius, M. Bouet, and J. Leguay, "Disco: Distributed multi-domain sdn controllers," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, May 2014, pp. 1–4.