Computer Science

Peter Dely

Adaptive Aggregation of Voice over IP in Wireless Mesh Networks

---

Master's Project

D2007:08

Adaptive Aggregation of Voice over IP in Wireless Mesh Networks

Peter Dely

This report is submitted in partial fulfilment of the requirements for the Master's degree in Computer Science. All material in this report which is not my own work has been identified and no material is included for which a degree has previously been conferred.

Peter Dely

Approved, 2007-06-07

Opponent: **Kerstin Andersson**

Advisor: **Andreas J. Kassler**

Examiner: **Donald F. Ross**

Abstract

When using Voice over IP (VoIP) in Wireless Mesh Networks the overhead induced by the IEEE 802.11 PHY and MAC layer accounts for more than 80% of the channel utilization time, while the actual payload only uses 20% of the time. As a consequence, the Voice over IP capacity is very low. To increase the channel utilization efficiency and the capacity several IP packets can be aggregated in one large packet and transmitted at once. This paper presents a new hop-by-hop IP packet aggregation scheme for Wireless Mesh Networks.

The size of the aggregation packets is a very important performance factor. Too small packets yield poor aggregation efficiency; too large packets are likely to get dropped when the channel quality is poor. Two novel distributed protocols for calculation of the optimum respectively maximum packet size are described. The first protocol assesses network load by counting the arrival rate of routing protocol probe messages and constantly measuring the signal-to-noise ratio of the channel. Thereby the optimum packet size of the current channel condition can be calculated. The second protocol, which is a simplified version of the first one, measures the signal-to-noise ratio and calculates the maximum packet size.

The latter method is implemented in the ns-2 network simulator. Performance measurements with no aggregation, a fixed maximum packet size and an adaptive maximum packet size are conducted in two different topologies. Simulation results show that packet aggregation can more than double the number of supported VoIP calls in a Wireless Mesh Network. Adaptively determining the maximum packet size is especially useful when the nodes have different distances or the channel quality is very poor. In that case, adaptive aggregation supports twice as many VoIP calls as fixed maximum packet size aggregation.

Keywords: Wireless Mesh Network, Voice over IP, Packet Aggregation

Acknowledgements

I would like to thank my supervisor *Andreas Kassler* for drawing my interest to wireless networking, suggesting the topic of this thesis and supporting me whenever questions arose, even outside of office hours. Without his help, this thesis would probably not have been written or finished. I am very grateful for all the support that I have received. Also, I would like to thank *Marcel C. de Castro*, who assisted me with various ns-2 problems, supplied the ETX-patch for ADOV-UU and gave me very useful suggestions for the design, implementation and testing of the aggregation algorithm.

I would like to thank my parents, who were the main funding source of my stay in Sweden. Without their support, it would not have been possible to study here. Last but not least, I would like to thank the girls from Duettgatan 5 for their constant supply of homemade chocolate cake and good mood.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Wireless Mesh Networks (WMNs) are multi-hop wireless networks in which the access points communicate with each other via wireless links. Thereby large areas can be covered with wireless access at low costs. In recent years, a lot of research into WMNs has been conducted. In 2007, the technology is advanced enough to hit the mass market. An IEEE proposal for WMNs[2] is in its final phase of standardisation and will soon be released. Large test-beds such as MIT's roofnet[3] have shown that the hardware and the protocols are mature enough for productive use. Currently first commercial deployments are made. For example the city of Isafjordur in Iceland recently took a wireless mesh into operation which provides cheap internet access to the remote surroundings of the city[4].

The use of IP networks for voice telephony became increasingly popular in the past years. In a *Voice over IP* (VoIP) system the speaker's voice is digitized and sent over an IP network in small packets. VoIP removes the requirement of having two separate networks for voice and data communications. This makes VoIP very flexible and cost efficient. Companies have started to replace their old PBXs and phones with VoIP enabled devices. VoIP software phones like Skype made VoIP also very popular among home users.

Combining both technologies is a very promising application. The ease of deployment of WMNs and the flexibility of VoIP contain a large cost saving potential. However, the first test calls in a standard WMN might be very disillusioning. VoIP over WMNs does not work very well, because the underlying wireless technology was not designed for real-time applications and the traffic characteristics of VoIP. Doing a VoIP call over a WMN can be compared with transporting passengers in an Airbus, while only conveying one passenger per flight. The Airbus represents the protocol overhead and the passenger the VoIP payload. The resources used for the overhead surpass the resources needed by the actual payload by far. As a result, the overall capacity of the airspace or the WMN is low. However, airlines want to make profit and therefore they try to use resources efficiently. So they try to transport as many passengers per flight as possible.

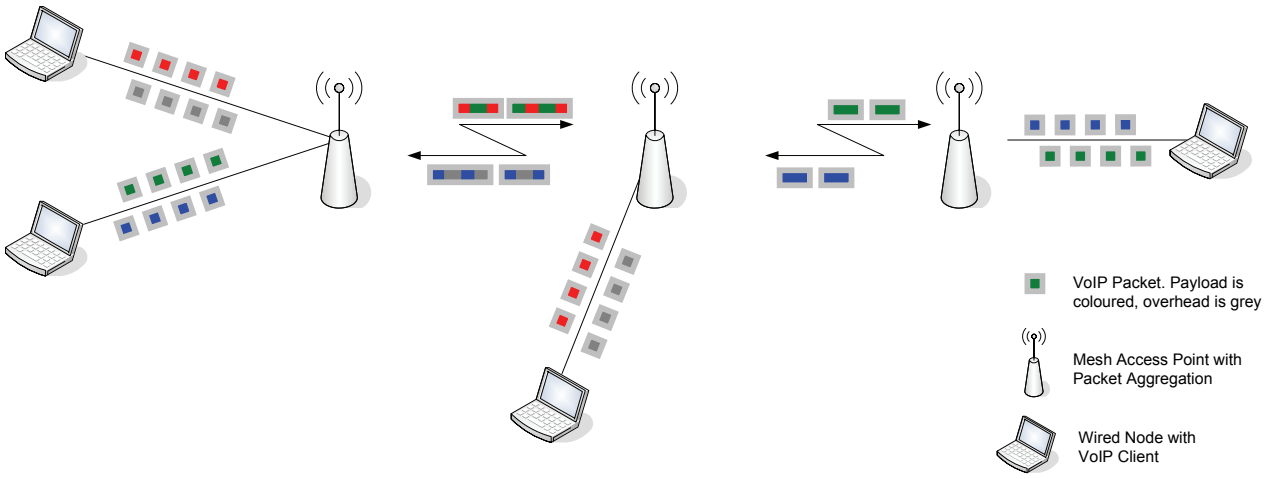

Figure 1.1 Voice over IP and Packet Aggregation in a Wireless Mesh Network

The same concept applied to VoIP is called *packet aggregation*. Many small packets are aggregated into a large one in order to enhance the overhead to payload ratio. Figure 1.1 shows the schematic operation of packet aggregation in a wireless mesh network. The small coloured rectangles symbolize VoIP packets. When they are transmitted over a wireless link, they are aggregated into larger entities. As a result, the overhead, symbolised by the grey border, is reduced.

1.2 Proposed Goals

The proposed goals of this master's thesis are:

- to give an overview of existing solutions for packet aggregation in Wireless Mesh Networks
- to develop a model for deriving optimum packet sizes for Wireless Mesh Networks taking into account link characteristics such as bit errors and packet loss rate
- to develop a novel scheme for packet aggregation in Wireless Mesh Networks, using the developed model and adapting the packet size to network traffic and link characteristics
- to implement this scheme in the network simulator ns-2[5]
- to evaluate the performance of the algorithm in different topologies focusing on Voice over IP traffic and to compare it to a non-adaptive algorithm

1.3 Document Outline

The rest of this paper is structured as follows:

- Chapter 2 provides background knowledge on Wireless Mesh Networks, Voice over IP and packet aggregation. It shows in detail what the problem of VoIP over WMNs is and why packet aggregation can reduce this problem.
- Chapter 3 proposes two new schemes for determining a packet size that enhances the capacity of a wireless mesh network. In addition, it defines design criteria for an aggregation algorithm.
- The implementation of one scheme combined with packet aggregation in the network simulator ns-2 is described in Chapter 4.
- In Chapter 5 the performance of this implementation is tested by simulation and compared with no aggregation. Also, this chapter explains how a realistic simulation environment can be set up in ns-2.
- Chapter 6 sums up and interprets the most important simulation results. In addition, enhancements of the presented scheme are suggested. Open questions for further research activities are proposed too.

2 Background

This chapter provides the necessary background information about Wireless Mesh Networks, Voice over IP and about packet aggregation to understand Chapters 3, 4 and 5.

2.1 Wireless Mesh Networks

2.1.1 System Architecture

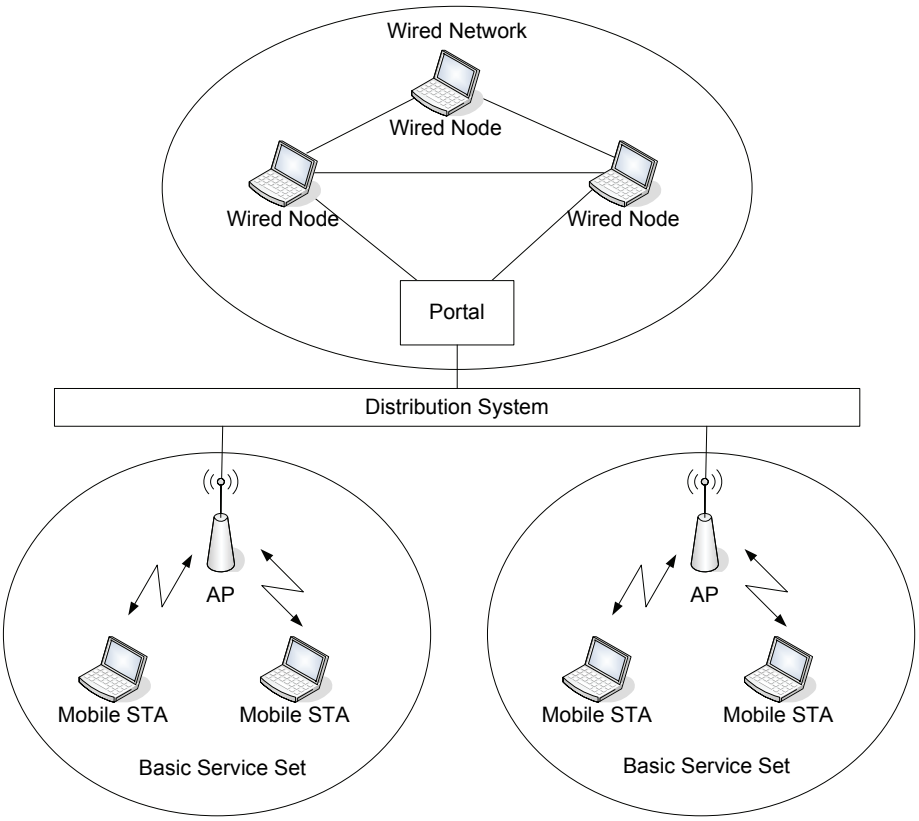

Figure 2.1 Extended Service Set. Source: [1]

In the recent years, wireless local area networks (WLANs) became very popular. The most common WLAN-standard is IEEE 802.11. This specification defines a physical and a Ethernet-like MAC-layer for wireless links[1]. The two types of nodes in an IEEE 802.11 WLAN are *mobile stations* (STAs) and *access points* (APs). A mobile station is a network device with a wireless network interface card, for example a notebook or a Personal Digital Assistant (PDA). An access point is a node connected to another network (e.g. a wired Ethernet) and acting as a bridge. Either the STAs communicate with each other to form an

independent basic service set (IBSS) or with the AP to form a *basic service set* (BSS). The APs are interconnected with wired links – the *distribution system* (DS). *Portals* are logical entry points for non-Ethernet packets[6]. All components together form an *extended service set* (ESS). An example of an ESS is displayed in Figure 2.1.

IEEE 802.11 networks are easy and cheap ways of providing wireless networking. However, the shortcoming is the wired connection between the APs. The wired link causes additional complexity and high deployment costs in certain situations[7]. Therefore it is desirable in many use-cases to connect the APs via wireless links as well and create a *WLAN Mesh*. In *Wireless Mesh Networks* (WMNs) APs turn into *mesh access points* (MAPs). Mobile stations are sometimes referred as *mesh clients*. The new IEEE 802.11s standard for WMNs introduces a third class of nodes called *mesh points (MPs)*[8]. MPs and MAPs support *WLAN mesh services*, allowing them to forward packets on behalf of other nodes to extend the wireless transmission range[9]. Mesh clients can associate with MAPs but not with MPs. *Mesh portals* are MAPs connected to a distribution system or a non IEEE 802.11 network. According to [9] there are three different types of WMNs (Figure 2.2):

- Infrastructure/Backbone WMNs: The MPs and MAPs form a meshed wireless network that serves as a backbone for the clients. The clients connect to the MAPs via standard 802.11, but do not forward packets. Sometimes different radios are used for client access and routing. The mesh routers can act as gateways to wired networks and other wireless technologies such as IEEE 802.16.
- Client WMNs: In this type the mesh clients form the network and no MAP is required. Mesh clients are – in contrast to MAPs - usually subject to energy constraints and incorporate mobility. Therefore client WMNs have other requirements than infrastructure WMNs. Client WMNs are also known as mobile ad hoc networks (MANETs). They should not be confused with legacy IEEE 802.11 ad hoc networks, in which STAs do not route traffic.
- Hybrid WMNs: This type is a combination of the first two. Here, both clients and MAPs are routers.

Subsequently the term “wireless mesh network” denotes the infrastructure variant.

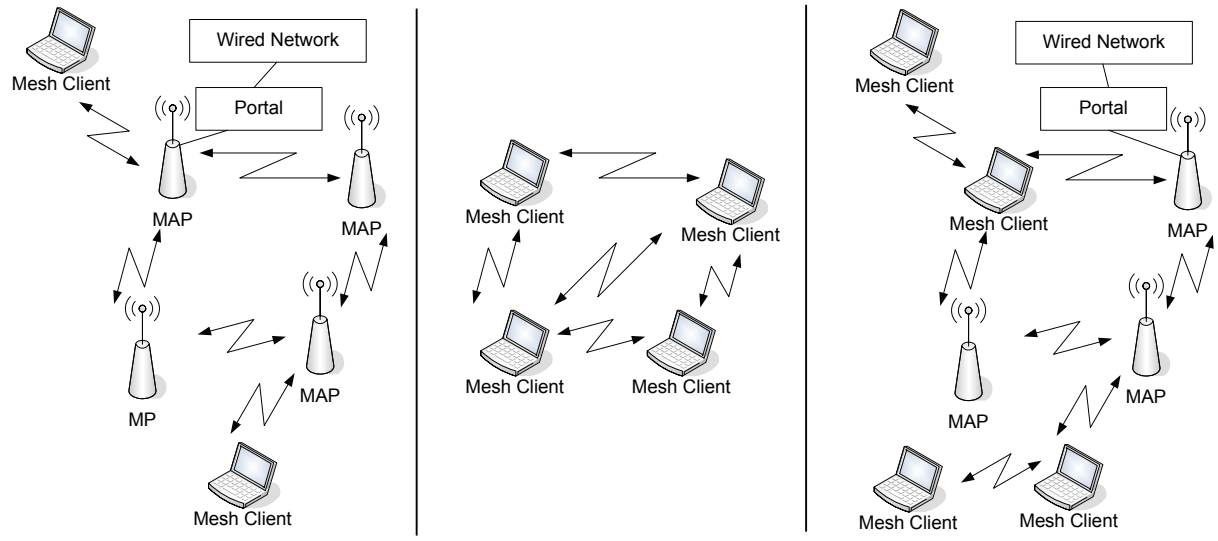

Figure 2.2 Architecture of a backbone WMN, client WMN and a hybrid WMN

Because of the system architecture mesh networks have different requirements to the physical layer, the MAC mechanism and the routing protocol than legacy IEEE 802.11 LANs. How these issues are handled in traditional IEEE 802.11 LANs and which changes have to be made for WMNs will be discussed in the rest of section 2.1. Also a short overview on typical application scenarios will be given.

2.1.2 Physical Layer

2.1.2.1 Introduction into the IEEE 802.11 Physical Layer

In IEEE 802.11 the physical layer is divided into the *physical layer convergence protocol* (PLCP) and the *physical medium dependent* (PMD) sublayer. The PLCP translates the MAC frames to a medium dependent format and provides a mechanism to the MAC layer to decide whether a channel is idle or not[6]. The PMD handles the signal modulation, encoding and decoding.

The original IEEE 802.11 standard supports two physical layers which are based on radio transmission and one which is based on infrared. The infrared physical layer is not used today; instead IrDA is the dominant standard for infrared communications[10]. For radio transmission either *frequency hopping spread spectrum* or *direct sequence spread spectrum* are used. These two techniques aim to maximize interference resistance and signal robustness.

When using frequency hopping spread spectrum (FHSS) the available bandwidth is split up into small frequency bands – 79 in IEEE 802.11 - with guard spaces in between[10]. The sender hops between the channels in a pseudo-random manner, whereas the sequence is both

known to the sender and the receiver. The use of many different narrow-banded channels increases interference resistance.

In direct sequence spread spectrum (DSSS) the original signal is XORed with a pseudo-noise – the chipping sequence. The resulting signal is spread over several channels. IEEE 802.11 uses a 11 chip Barker sequence, spreading the 1 MHz input signal to 11 MHz[10]. Damaged transmissions can be recovered by statistical methods[6].

Beside the spread spectrum mechanism the modulation scheme is an important attribute of the physical layer. IEEE 802.11 uses *Gaussian frequency shift keying* (GFSK) in FHSS and *phase shift keying* (PSK) in DSSS. Data rates of 1 and 2 Mbps are possible. By using complimentary code keying IEEE 802.11b supports higher transmission rates of 1, 2, 5.5 and 11 Mbps. The transmission rates can be adapted to the link quality. To further increase the throughput a new physical layer was introduced in 802.11a/g, which supports up to 54 Mbps. IEEE 802.11a/g uses *orthogonal frequency-division multiplexing* (OFDM). Here the carrier is split up into 48 subcarriers, which each transmit a part of the input stream at a lower bit rate[10]. Lower rates are less vulnerable to interference. The newest standard IEEE 802.11n will use OFDM in combination with multiple antennas. Thereby data rates of more than 100 Mbit/s will be possible.

2.1.2.2 Physical Layer in WMNs

To increase physical layer performance in WMNs new antenna systems have been proposed[9]. The use of multiple antennas at the sender and the receiver to create a multiple-input multiple-output (MIMO) system can triple the performance. Spatial division through directional antennas is an option for WMNs. However, these antenna systems increase the hardware costs and also require new MAC protocols. IEEE 802.11s does not include a physical layer specification[8].

2.1.2.3 IEEE Standards for WLAN

A few IEEE 802.11 standards on wireless networks have been mentioned above. A comparison of the most used standards is provided in Table 2.1. They differ in data rate, frequency, the physical layer and the MAC layer. In Europe the most common standards are 802.11b and 802.11g, which operate at the freely available 2.4 GHz ISM band.

Standard	802.11	802.11a	802.11b	802.11g	802.11n
Max. Data Rate	2 Mbps	54 Mbps	11 Mbps	54 Mbps	> 100 Mbps
Frequency	2.4 GHz	5 GHz	2.4 GHz	2.4 GHz	2.4 or 5 GHz
Physical Layer	FHSS/DSSS	OFDM	DSSS	OFDM	OFDM MIMO
MAC Layer	CSMA/CA	CSMA/CA	CSMA/CA	CSMA/CA	CSMA/CA

Table 2.1 Comparison of wide-spread 802.11 standards, Sources: [1, 6, 11, 12]

2.1.3 MAC Layer

2.1.3.1 Introduction into Media Access Control

The radio channel is a common resource, which all participants in a wireless network share. To provide fair access to this resource and to utilize bandwidth efficiently a *media access control* (MAC) mechanism is necessary[1]. Since it has been proved [10] that allowing media access to discrete instants of time only improves efficiency, MAC protocols also take care of network wide time synchronisation. Beside regulating the access to the channel and time synchronisation the IEEE 802.11 MAC layer is also responsible for system authentication, (de)association with an access point, encryption and data delivery[6]. This section will only describe the medium access control mechanisms.

In wired Ethernets the MAC scheme is *carrier sense multiple access with collision detection* (CSMA/CD). When a station wants to send, it senses the medium (the cable) to check if it is idle. If the medium is idle, the station starts the transmission. If the medium is busy, the station continues sensing the medium until it is free and then begins sending[10]. It can happen, that two stations begin sending at the same time. As a consequence the signals collide on the medium. A sending station can detect such a collision by comparing the signal on the channel with the signal it sends to the channel. If both signals are different the station has detected a collision, aborts sending and sends out a jamming signal to all other stations[1]. After a random backoff interval, the station tries sending again.

CSMA/CD works well in wired networks. However it can not be used in wireless networks, because wireless communication has some unique properties [13]: first, the wireless transceivers are usually half-duplex, to avoid self interference when sending and receiving at the same time. Therefore, collision detection cannot be performed. The second unique property is the change of received signal quality over time due to reflection, diffraction and scattering of radio waves. Consequently, the third property is a bit error rate that is about thousand times higher than in wired networks. The fourth attribute of wireless

communication is the location dependent carrier sensing. Since a sent signal degenerates over distance, it can only be detected within a specific radius around the sender. This property causes the *hidden terminal problem* and *exposed terminal problem*[1, 10].

The hidden terminal problem can be illustrated by the following example (see Figure 2.3): S1 sends data to R1. S2 wants to send data to R1 at the same time. It senses the medium idle, since it is out of S1s communication range. Therefore it starts sending. The signals of S1 and S2 collide at R1. In this example S1 is a *hidden terminal* to S2. It can be noticed, that the collision occurs at the receiver, not at the sender.

The second problem which is unique for wireless communications is the exposed terminal problem. Here S1 sends data to R1. S3 wants to transmit to R2, but as it hears the communication between S1 and R1 it concludes that the medium is busy and waits for the medium to become idle. Nevertheless, S3 could send in this situation, since R1 is out of its transmission range and no collision would occur at R1. In this example S3 is *exposed* to S1.

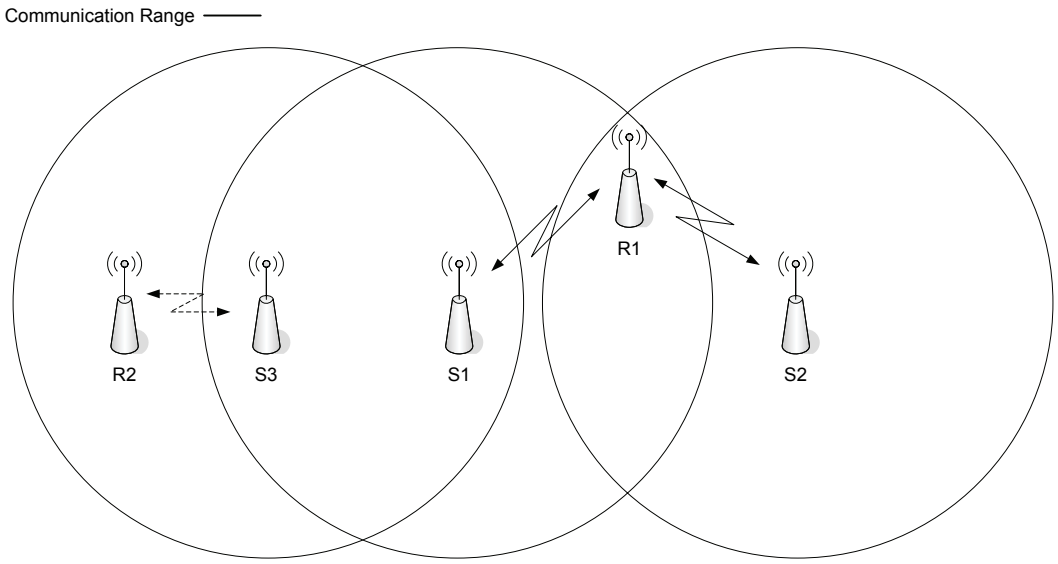

Figure 2.3 Hidden and exposed terminals

The hidden and exposed terminal problem illustrate why CSMA/CD is not applicable in wireless networks. Sections 2.1.3.2 and 2.1.3.3 will describe MAC protocols for single-hop WLANs and then challenges and solutions for MAC protocols in WMNs.

2.1.3.2 MAC Protocols for single-hop WLANs

IEEE 802.11 supports three MAC protocols. For networks with an access point the *point coordination function* (PCF) is available. The two protocols for ad hoc networks are summarized as *distributed coordination function* (DCF)[10]. All three protocols are *slotted*

protocols where time is split into slots of 20 µs (FHSS) or 50 µs (DSSS) and transmissions are synchronized to this time slots[10, 13]. Different inter-transmission waiting times allow prioritization of messages. IEEE 802.11 defines:

- short inter-frame spacing (SIFS): 10 µs (DSSS) or 28 µs (FHSS)
- PCF inter-frame spacing (PIFS): SIFS + one slot time
- DCF inter-frame spacing (DIFS):PIFS + two slot times

Distributed Coordination Function with CSMA/CA

Carrier sense multiple access with collision avoidance (CSMA/CA) works similar to CSMA/CD, but does not use collision detection for the previously mentioned reasons. The process of sending a unicast data frame is shown in Figure 2.4. When a station wants to send, it senses the medium. If the radio channel is idle for at least the duration of DIFS, the station sends. If the medium is busy it waits until it is idle for DIFS. Then it enters a contention phase choosing randomly a multiple slot-time within a *contention window* (CW) as a backoff timer. If the medium is idle when the timer expires, the station transmits. If the medium is busy before the timer expires, the station stops the timer and starts it again after the channel is idle for DIFS[10]. After each unsuccessful transmission attempt the size of the CW size is doubled. As a consequence, the waiting time increases and the probability of a concurrent transmission decreases. As a result of this scheme the delay is increased when a high network load is present. When a station receives a unicast frame it waits for the duration of SIFS (which is shorter than DIFS) and sends back an *acknowledgement* message (ACK). After a successful transmission the sender resets the CW size[14].

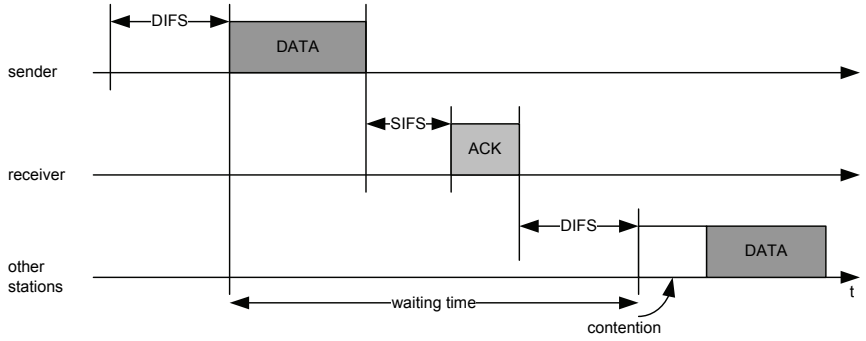

Figure 2.4 IEEE 802.11 unicast data transfer. Source: [10]

Distributed Coordination Function with RTS/CTS extension

CSMA/CA does not overcome the hidden terminal problem, since collisions can still occur at the receiver. Therefore an extension to DCF was made, introducing two new control packets and a virtual channel reservation scheme[10, 14].

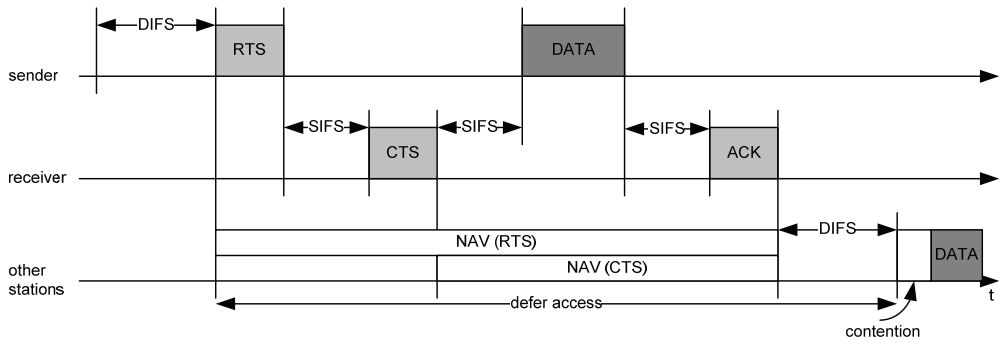

Figure 2.5 IEEE 802.11 unicast data transfer with RTS/CTS. Source: [10]

As displayed in Figure 2.5, a station waits for the duration of DIFS and then sends a *request to send message* (RTS). This includes a duration field specifying the expected time needed for the transmission of data and acknowledgement. Every node overhearing the RTS stores the medium allocation in his *net allocation vector* (NAV). The receiver waits for SIFS and sends a *clear to send* (CTS), also including the duration field. Again, all nodes overhearing the CTS, set their NAV. From now the channel is reserved for the sender. He waits for SIFS and transmits the data. The receiver responds with ACK after waiting SIFS.

In this scheme the hidden terminal problem still exists for RTS messages[13]. Usually RTS is much smaller than the data frame, therefore RTS collisions are not as severe. [15] claims that for certain topologies RTS/CTS is very ineffective. For small data frames it is beneficial to omit RTS/CTS[10], since the overhead surpasses the benefits of the handshake.

Two protocols similar to DCF with RTS/CTS are the *Media Access Collision Avoidance Protocol* (MACA) and the *Media Access Collision Avoidance protocol for Wireless LANs* (MACAW). MACA utilizes a RTS/CTS handshake, but sends no ACK upon a successful transfer[1]. MACAW uses a RRTS-RTS-CTS-DS-DATA-ACK exchange mechanism. The receiver sends a request-for-request-to-send message (RRTS) to the sender, whenever he could not respond to a previous RTS because of contention in his range. The sender then triggers the RTS/CTS process. The data-sending message (DS) is sent before the actual data transmission and tells exposed hosts that the RTS/CTS was successful[1].

PCF

Using the point coordination function (PCF) transmission time is assigned to the stations by the access point who acts as a *point coordinator* (PC). The PC splits up the access time into super frames, which comprise of a beacon frame, a contention-free period and a contention period. After sending the beacon frame the PC polls the associated stations consecutively. When a station has data to send, it waits SIFS and begins the transmission. If not, the PC

waits for PIFS (which is longer than SIFS) and polls the next station. To end the contention-free period and start the contention period the PC broadcasts a CF-End message[10]. Within the contention phase the stations use one of the DCF operation modes to register at the base station. To start a new contention-free phase the PC has to gain media access using the DCF. Therefore the start of the contention-free period can be delayed. As a result PCF can not provide Quality of Service (QoS) support[1].

QoS enabled versions of PCF and also DCF are specified in IEEE 802.11e[12]. In this standard *service differentiation* is implemented with traffic classes or access categories, which use inter-transmission waiting periods of different lengths[1].

2.1.3.3 MAC Protocols for WMNs

Wireless mesh networks are distributed multi-hop wireless networks. Because of the distributed nature centralized MAC protocols like PCF cannot be used. Distributed protocols such as DCF or MACA can be used. However they show bad performance, do not solve the hidden and exposed terminal problem and do not provide fairness[16, 17]. As a consequence MAC protocols for single-hop WLANs do not work properly in WMNs[9]. Also protocols for multi-hop ad hoc WLANs do not meet all requirements for WMNs MAC protocols, since WMNs have different properties than ad hoc networks[18]. For example, they do not comprise of routers but also stations, the routers incorporate low to zero mobility and are not subject to energy consumption constraints. Consequently specialised MAC protocols should be used in WMNs.

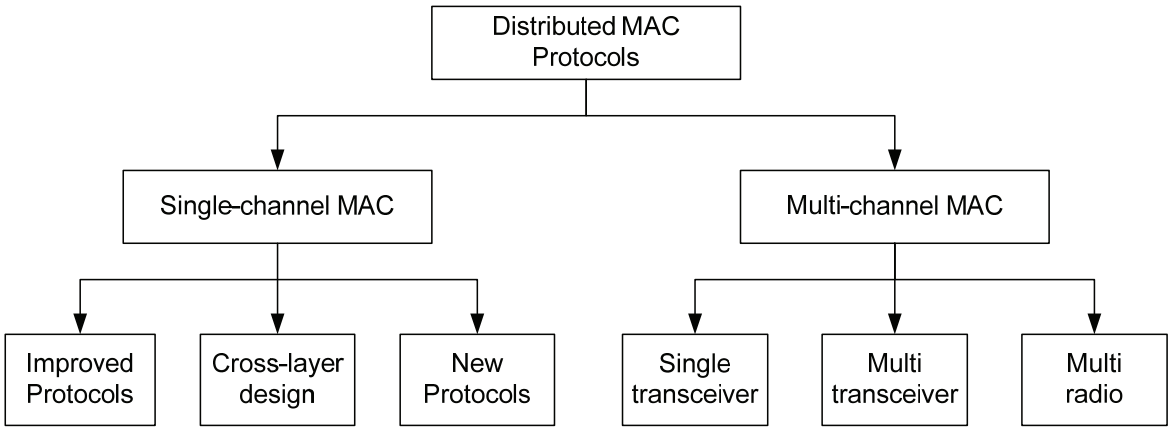

Figure 2.6 Classification of distributed MAC protocols

Different ways of implementing distributed MAC protocols for WMNs are shown in Figure 2.6[9]. In one group the nodes use a single channel for transmissions, in the other they

can use multiple channels[9, 18]. MAC protocols for both scenarios will be presented in the following paragraphs.

Single-channel MAC protocols

One way to adapt legacy MAC protocols for WMNs is to change single parameters like CW size or back off procedures. However, it has been shown that no major performance gains can be achieved[9]. Another approach is to combine physical layer enhancements such as directional antennas with new MAC protocols. The MAC protocol must be able to cope with the intensified hidden node problem introduced by the directional antennas. Also it is an alternative to let the MAC protocol regulate the sending power.

New introduced WMN MAC protocols mainly try to provide QoS mechanisms and enhance fairness[18]. QoS can be achieved with traffic prioritization and resource reservation. Fairness is a major problem for multi-hop WLANs. Traditional MAC protocols provide per-hop fairness, but fail in multi-hop WLANs. In a WMNs with DCF as MAC protocol one TCP session can shut down an already established session[17].

Multi-channel MAC protocols

Multi-channel protocols aim for a more efficient utilization of the radio spectrum. According to [9] there are three different ways to implement multi-channel operation. The first option is *multi-channel single-transceiver* MAC. Here the nodes have one transceiver, that can switch to multiple channels, but send only on one at a time. The MAC protocol selects transmission channel. The second variant is *multi-channel multi-transceiver* MAC. Here the nodes have multiple transceivers that can use several channels simultaneously, but only one MAC protocol on top to coordinate the transmissions. In contrast, the third variant – *multi-radio* MAC – uses an own MAC protocol for each of its independent radios and puts a virtual MAC protocol on top. Multi-channel single-transceiver MAC is easiest to implement, since no additional hardware for multiple transceivers is required.

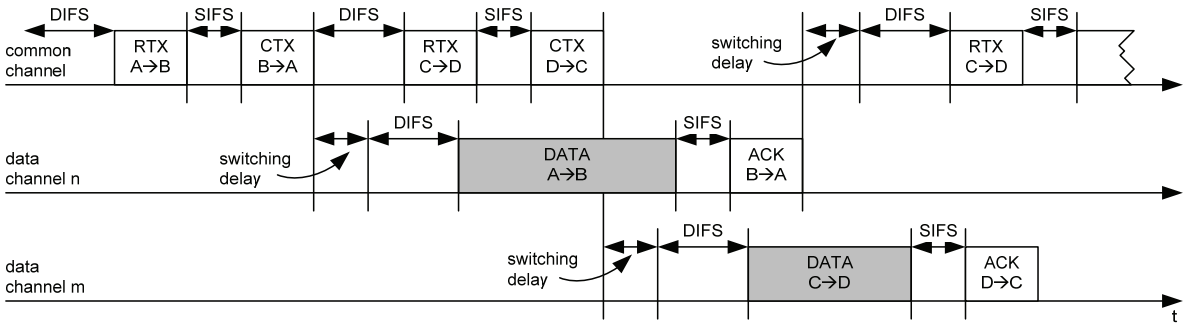

Figure 2.7 Common Channel Framework in IEEE 802.11s, Based on:[8]

The upcoming wireless mesh network standard IEEE 802.11s will support an optional multi-channel single-transceiver MAC protocol called *Common Channel Framework* (CCF)[8]. The sender transmits a Request-to-Switch (RTX) message (instead of RTS), which includes a suggested channel number, on a common channel to the receiver. The receiver responds with a Clear to Switch (CTX) (instead of CTS), accepting or declining the channel selection. After a successful RTX/CTX handshake sender and receiver switch to the negotiated channel, which causes the channel switching delay. Then the sender transmits the data and the receiver responds with an ACK. Afterwards both tune back to the common channel. Figure 2.7 shows this process for two sender-receiver pairs A/B and C/D. Due to the use of separate channels the ACK from A to B can be transferred in simultaneously with the data from C to D.

To increase the utilization of the common channel a channel coordination window (CCW) is defined, in which the common channel is solely used for RTX/CTX. Outside the CCW the common channel can also be used for data transfers. The CCW is repeated after a defined period.

2.1.3.4 Summary

Wireless mesh networks require a new class of MAC protocols, for example improved single-channel protocols or multi-channel protocols. Although legacy protocols like IEEE 802.11 DFC or MACA can be used, specialized WMN MAC protocols show better performance. However, a lot of research into the design of WMN MAC protocols is currently done and new solutions are likely to emerge in the coming years.

2.1.4 Routing Protocols

2.1.4.1 Introduction into Routing Protocols

Wireless mesh networks are multi-hop networks. Therefore a mechanism for finding a path between source and destination is needed. *Static routing* means that the path is set up manually, while *dynamic routing* requires a *routing protocol* which sets up *routing tables*. A router forwards packets to a next hop neighbour, which is chosen upon a routing metric. This process is called *routing[19, 20]*. In this section only protocols for one-to-one communications, so called *unicast routing protocols,* will be covered.

In traditional wired networks either *distance-vector protocols* or *link-state protocols* are used. With distance-vector protocols each node keeps a vector of its neighbours and their distance. The router periodically broadcasts the distance-vector to its one-hop neighbours[20].

Those use the Bellman-Ford algorithm to determine the distance to all neighbours. Distance is usually the hop-count, sometimes it can be a weighted hop-count, that takes information like link capacity into account[19]. RIP and BGP are well-known distance-vector protocols. In contrast link-state protocols send updates whenever the topology changes. Routers build a map of the network and use the Dijkstra's algorithm to calculate the shortest path to a destination. OSPF and IS-IS are wide spread link-state protocols.

A different classification of routing protocols is useful for WMNs[1]. *Proactive routing protocols* maintain routes to all hosts in the network, while *reactive routing protocols* only find a route when one node wants to communicate with another. *Hybrid routing protocols* are a combination of the first two. Here some routes to destinations, e.g. in the vicinity of a node are maintained proactively and others, e.g. routes to far away distances, are created on demand.

In WMNs either *source-routing protocols* or *hop-by-hop routing protocols* are used[21]. Source routing protocols specify the path a packet has to follow in each packet header. Consequently, the relaying nodes do not need routing tables and a high routing overhead is generated for small packets. Hop-by-hop routing uses routing tables to determine the next hop on each router.

Two major attributes of routing protocols are *efficiency* and *convergence time*. Efficiency is the share of routing traffic in overall traffic. Proactive routing protocols are efficient in high load scenarios while reactive protocols work well in low traffic settings[7]. Convergence time refers to the time span needed to have correct routing tables after a topology change. Link state protocols usually have a better convergence time than distance vector protocols.

2.1.4.2 Routing in Wireless Mesh Networks

Routing protocols for Wireless Mesh Networks are different from conventional routing protocols in several ways. A very significant difference is that some WMN routing protocols such as the *hybrid wireless mesh protocol* (see section 2.1.4.2) operate on layer 2, while traditional routing protocols are layer 3 protocols. Layer 2 routing is necessary because Mesh Access Points and Mesh Points do not have IP capabilities. Also, *multi-radio routing* for capacity improvement and *multi-path routing* for load balancing and fault tolerance can be desirable in WMNs. The next three subsections give an overview on routing metrics of WMNs and describe two routing protocols.

Routing Metrics in WMNs

To decide if one route is better than another one *routing metrics* are required[19]. As shown in [21] and [22] the selection of an appropriate routing metric affects the network performance significantly. Traditional routing protocols use link cost or hop-count as the route selection criterion. However, these metrics are not suitable for Wireless Mesh Networks. Link cost is manually assigned to each link, which is not possible in self organising WMNs. Minimizing the hop-count does not necessarily lead to a good path, since a two-hop path over two stable links can be better than a single lossy link[22]. [21] proposes the use of the following metrics for WMNs:

- Expected Transmission Count (ETX): *estimates the number of retransmissions needed to send unicast packets*[22]. It is calculated by broadcasting probe messages every second to neighbouring nodes and counting the received probe messages within the last ten seconds. The number of received probe messages is piggy-packed on the probe messages. The ETX is calculated as $ETX = \frac{1}{d_f * d_r}$, where d_f denotes the delivery ratio of packets in the forward path direction and d_r the delivery ratio in the reverse direction[23].

  Example: 9 out of 10 probe messages from A were received by B and 8 out of 10 from B were received by A, i.e. the loss ratio is 0.1 for A-B and 0.2 for B-A. The probability of a successful data transfer from A-B and an acknowledgement is $(1-0.1)*(1-0.2)=0.72$. The ETX for the link A-B is therefore $1/0.72 \approx 1.39$.

- Expected Transmission Time (ETT): estimates the MAC layer duration needed for successfully transmitting a packet[21]. The ETX for one link is calculated as $ETT = ETX \frac{s}{b}$, where s is the packet size and b the transmission rate of the link. The ETT is especially interesting when link speeds vary from hop to hop, for example by the use of different standards such as IEEE 802.11b and IEEE 802.11g[24].
- Weighed Cumulative ETT (WCETT): *reduces the number of nodes on the path of a flow that transmit on the same channel[25]*. The WCETT is not isotonic. Isotonicity means that the order (length) of two paths remains when a common third path is appended or pre-fixed[21] Without this property, common algorithms cannot calculate the minimum weight path.
- Metric of Interference and Channel Switching (MIC): is similar to WCETT, but fulfils the property of isotonicity.

[21] and [22] show that ETX, ETT and MIC outperform hop-count in Wireless Mesh Networks.

Ad hoc On-demand Distance Vector Routing (AODV)

AODV is a reactive hop-by-hop routing protocol developed for wireless ad hoc networks[26]. When a host wants to find a route to a destination it broadcasts a route request (RREQ) message. The RREQ includes source and destination address, source and destination sequence number and a broadcast identifier. When an intermediate node receives a RREQ, it either re-broadcast the request or answers with a route reply (RREP) if it has a fresh enough route.

When forwarding a RREQ the node stores broadcast identifier, source address and the reverse route. RREQ are only re-broadcasted when a request with the same source address and broadcast identifier has not been processed before. This avoids loops of RREQ messages. If the node's destination sequence number is smaller than the requests destination sequence number, it sets it to the requests sequence number. Thereby the destination sequence number is an indicator of route freshness. An intermediate host replies with a RREP when it has a fresh enough route to the destination.

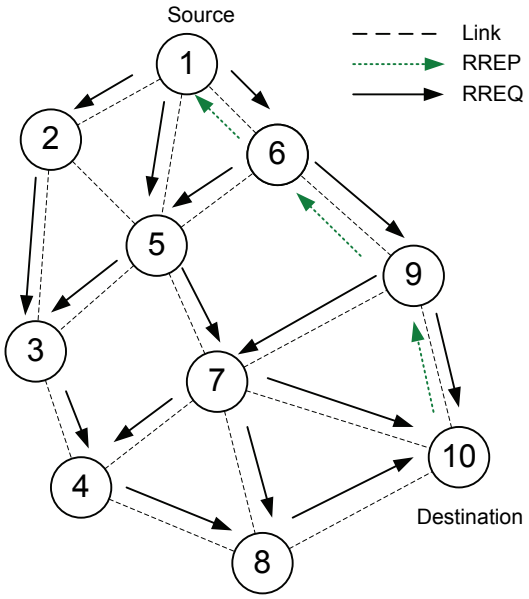

Figure 2.8 AODV route discovery example

Figure 2.8 shows an example of an AODV route discovery process. Node 1 wants to find a route to node 10. It broadcasts a RREQ, which is re-broadcasted by nodes 2, 5, 6 etc. When forwarding this RREQ the nodes set up a reverse route. Finally the RREQ reaches the destination, node 10. This node unicasts a RREP to node 1.

AODV does not comprise a local link repair mechanism. Instead, ADOV monitors the link status to its direct neighbours either by periodic HELLO messages or link layer acknowledgements[26]. When a link failure is detected the intermediate nodes send route error messages to the end node. For example if the link between node 6 and node 9 breaks, both invalidate the route and send out route error messages to node 1 respectively node 10. Node 1 starts a new route discovery process with a new broadcast id and the old destination sequence number[1].

Hybrid Wireless Mesh Protocol (HWMP)

HWMP is the routing protocol of IEEE 802.11s. It is a layer 2 routing protocol which uses MAC addresses instead of IP addresses. IEEE 802.11s also denotes it as a *path selection protocol* instead of routing protocol, to make this difference clear. HWMP is a hybrid routing (path selection) protocol, which uses the *Radio Metric Ad hoc On-Demand Distance Vector* (RM-AODV) protocol for proactive routing and tree based routing for reactive routing[8, 27]. RM-AODV is an enhanced AODV version, which allows arbitrary metrics for path selection, uses an optimized RREQ/RREP mechanism and optionally performs proactive route maintenance to popular hosts. Radio Metric indicates that the routing metrics take the quality of the radio link into account.

HWMP supports two operation modes. If no *root portal* is configured, RM-AODV is used for path selection. For destinations within the mesh the route discovery works like normal AODV. If the destination is outside the mesh (e.g. on a wired LAN), the source receives no RREP upon a RREQ. Therefore it sends the messages to the route portal after a timeout. The portal forwards them to the connected LAN[27].

If a root portal is configured (e.g. node 1 in Figure 2.9) the nodes proactively maintain a path to the root portal. For example node 10 wants to communicate with node 3. It first checks if it has a route to node 3. If not it sends the message to the root portal on the proactive path. The portal forwards it to node 3. Node 3 can respond via the proactive path or establish a direct path using RM-AODV. If node 10 wants to communicate with a host on the LAN it directly sends the message to the portal, which forwards it to the LAN[27].

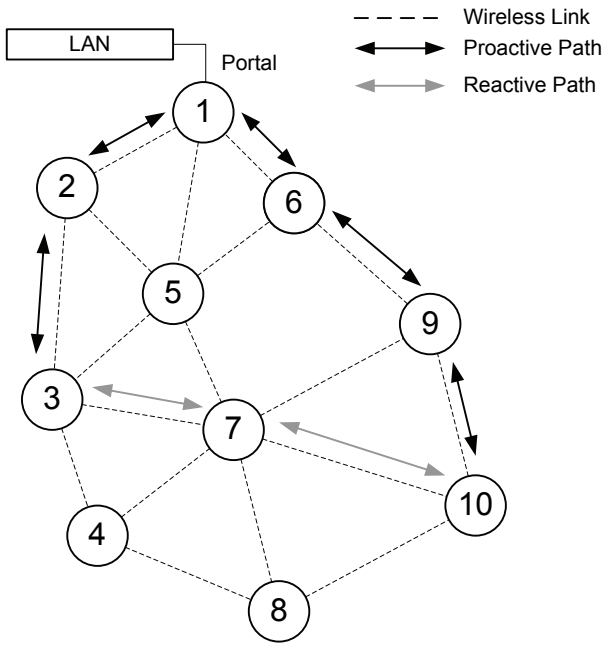

Figure 2.9 HWMP example

2.1.4.3 Summary

Wireless mesh networks need new routing protocols and routing metrics. AODV is not ideal for WMNs, since it uses hop-count as a routing metric[28]. Also layer 3 routing does not fit into the concept MAPs, which are layer 2 devices. HWMP eliminates these shortcomings.

2.1.5 WMN Application Scenarios

The ability to cover large areas with cheap hardware and a self-organizing nature creates a number of interesting application scenarios for WMNs, which are not possible with legacy IEEE 802.11 LANs or cellular networks. Some of the applications suggested in [9] and [2] are:

- Community/public access networking: WMNs can be used to share internet access and provide location based services. The WMN can cover limited areas like university campuses or even whole cities. Projects like freifunk.net[29] aim to build up non-commercial roof top networks for providing free internet access.
- Broadband home networking: WMNs eliminate the difficulties of adding new APs which need wired network access. WMNs can serve as a distribution system for internet access and multimedia streams.
- Enterprise Networking: WMNs can be used where no wired Ethernet is available or where an installation is too expensive.

- Emergency Networking: emergency response teams can use WMNs for voice and data communication at an incident site.

As wireless mesh networking is a new technology and protocols are still subject to standardization efforts, it is very likely that the industry will develop new application scenarios and appropriate products in the coming years.

2.2 Voice over IP

2.2.1 Overview

The International Telecom Union (ITU) defines Voice over IP (VoIP) as *the transmission of voice, fax and related services over packet-switched IP-based networks*. Services that were traditionally provided by public switched telephone networks systems (PSTN) are run over IP based networks such as LANs or the Internet. The terminals, i.e. the phones, are IP enabled devices such as computers or IP phones.

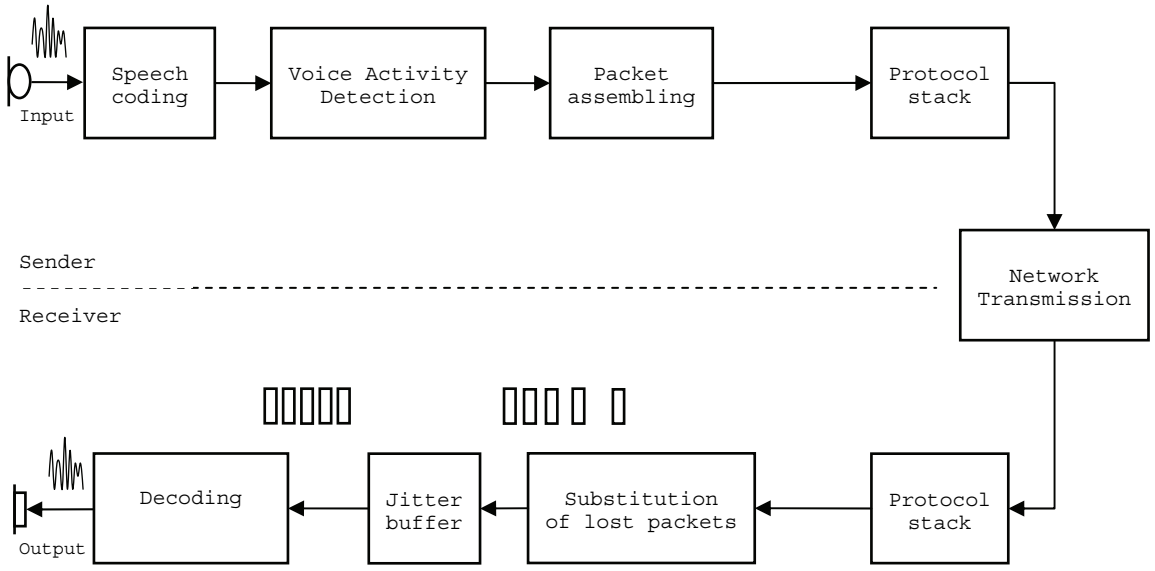

Figure 2.10 Processing of audio signals in a VoIP system

Figure 2.10 displays the setup of a VoIP system and how audio signals are processed by it[6]. At the sender the digitized voice of the speaker is encoded by a speech encoder, then packed and sent through the protocol stack. If the speaker is silent, the voice activity detection (VAD) recognizes this and a packet without payload or no packet is generated. Afterwards the packets are sent over the network, for instance an IP-based LAN. At the receiver side the protocol stack processes the packets. Then lost packets are detected and substituted. Since the packets might not arrive in a constant flow, they are collected in a jitter buffer that adjusts

time differences and the arrival order. Finally, the decoder decodes the packets and outputs them via the sound system.

There are a number of standards used for signalling, speech coding and the transport of the voice packets over IP. This modularization makes VoIP flexible and the standards interchangeable when new requirements and applications emerge. The following sections will give an introduction in the most important standards and issues related to VoIP.

2.2.2 Standards and Protocols

2.2.2.1 Audio Codecs

The input to and output of a VoIP system are analogue waveforms. To transmit these waveforms over the digital network, they need to be converted to a digital format at the sender and converted back to an analogue signal at the receiver. The system doing this conversion is called codec, a combination of coder and decoder[30]. According to [31] bit rate, delay, complexity and quality are the most important attributes of codecs.

The most common codecs for VoIP are described underneath and in Table 2.2. The bit rate defines how much bandwidth is needed to transmit one encoded stream (without protocol overhead). The frame size denotes how much data is processed by the encoder at once. Sometimes a look ahead buffer in the next portion of input is used to optimize the encoding. These two aspects generate delay (also see section 2.2.3.1). In the table the impairment factor (also see section 2.2.4.2) is given as a quality metric. Complexity is a measure of the number of CPU instructions and the amount of RAM needed.

Attribute	G.711	G.723.1	G.729 Annex A
Bit rate	64 kbit/s	5.3/6.3 kbit/s	8 kbit/s
Delay			
Frame size	20 ms[1]	30 ms	10 ms
Look ahead	0 ms	7.5 ms	5 ms
Quality (Ie value[2])	0	15	11
Complexity			
MIPS	n/a	16	20
RAM	n/a	2200 words	3000 words

Table 2.2 Comparison of Codecs, Sources: [31, 32]

[1] Using the RFC 3551 recommendation to send 20ms of G.711 encoded audio in one packet.
[2] Impairment factor at zero packet loss.

G.711

One of the oldest codecs is G.711[31], which uses Pulse Code Modulation (PCM) for encoding. The encoder polls the input signal at a rate of 8000 Hz. The energy level of each sample is mapped to an 8-bit value. Therefore the resulting bit rate is 64 kbit/s[30]. By default 20 ms of PCM-samples are sent in one packet[33]. That means that 20 ms of delay are added by the encoder and packetization respectively. G.711 has high bandwidth requirements, low delay, low complexity and the best voice quality of the three standards[32].

G.723.1

The G.723.1 codec is designed to achieve high audio quality on low bandwidth links while having a "limited amount of complexity"[34]. The encoder takes 240 16-bit PCM samples, which are generated at a sampling rate of 8000 Hz, as input. It then uses Multipulse *Maximum Likelihood Quantization* to generate output with 5.3 kbit/s or *Algebraic-Code-Excited Linear-Prediction* for output with 6.3 kbit/s rate. The encoder generates frames of 30ms length and has a look ahead buffer of 7.5 ms. Therefore the algorithmic delay introduced by G.723.1 is 37.5 ms. The quality of G.723.1 is worse than the quality of G.711 or G.729[32].

G.729

The G.729 coder outputs frames of 10 ms, which corresponds with 80 16-bit PCM samples at a sampling rate of 8,000 Hz. The output bit rate is 8kbit/s. The encoder utilizes *Conjugate-Structure Algebraic-Code-Excited Linear-Prediction* for encoding[35]. Since a look ahead buffer of 5 ms is part of the encoder, the algorithmic delay is 15 ms. Usually two frames are sent in one packet[33]. The coding scheme used in G.729 is rather complex. Therefore, a simplified version of the coder is introduced in Annex A. This codec has a slightly lower quality, but does not require as much memory and CPU power[31].

2.2.2.2 Signalling Protocols

A VoIP call is a stream of digitized audio over an IP network. Setting up this media stream requires a signalling protocol, which determines how the caller and the callee communicate with each other[6]. In addition, signalling protocols are responsible for a number of other issues, such as termination of a call or registering a terminal at a central address server. In the currently most popular signalling protocols, *H.323* and *Session Initiation Protocol* (SIP), signalling is independent from the media flow (out-of-band signalling).

H.323

The H.323 protocol family is the ITUs specification for the transmission of audio, video and data over IP networks[36]. It is an umbrella standard that references other standards for call

control, network transport and audio encoding. H.323 uses the TCP-based H.225 protocol for call control and the UDP-based Real Time Protocol (RTP) for network transport (also see section 2.2.2.3). The specification also brings along a list of voice and video coding standards. The long list of H.323 sub-standards makes the clients complex[37].

Session Initiation Protocol

The session initiation protocol (SIP) is an alternative to H.323, which is promoted by the Internet Engineering Task Force (IETF). It covers roughly the same functionality as H.323[37]. SIP is clear-text based and uses HTTP-like messages for signalling. SIP is lower-layer neutral, i.e. the data transport protocol is independent from SIP. However, RTP is used in most cases. Also the audio codecs are not part of the standard[38].

2.2.2.3 Data Transport Protocols

The voice frames generated by the audio codec are transported over the network using RTP over UDP over IP (see Figure 2.11). The User Datagram Protocol (UDP) is a connectionless protocol that includes – in contrast to TCP - no error correction mechanism. It is used for VoIP because of its lower overhead compared to TCP. Moreover, there is no need for a reliable connection for transporting voice, as a certain packet loss can be tolerated and handled by the audio codec (see section 2.2.3.2).

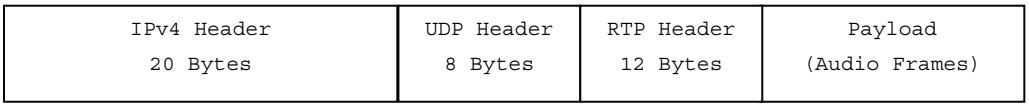

Figure 2.11 Encapsulation of VoIP Traffic

The UDP header does not include sequence numbers or timestamps. Since those fields are needed for VoIP the Real Time Protocol (RTP), which is specified in Request for Comment (RFC) 1889[39], is used in the layer above UDP[6]. The RTP header is 12 bytes long and includes a timestamp, a sequence number and a packet type field. The timestamp is used to calculate the packet delay. The sequence number is needed to restore the correct order of the packets. The packet type field gives information about the used codec[30]. The RTP payload is the actual audio sample, encoded by some audio codec. The RTP header does not include a payload length field, since this can be calculated from the UDP length field.

Along with RTP comes the RTP Control Protocol (RTCP). Beside the RTP session, the communication partners establish a RCTP session to exchange information about the link quality, such as delay, packet loss and jitter.

As shown in Figure 2.11 the IP/UDP/RTP overhead is 40 bytes per packet. To save bandwidth VoIP applications sometimes use *RTP header compression*. Header compression makes use of the fact that most header fields only change little (e.g. timestamps) or stay static (e.g. source address) during a transmission. Different mechanisms are available, whereas Robust Header Compression (ROHC)[40] is also applicable over wireless links. ROHC can reduce the overhead to one byte per packet.

2.2.3 Characteristics of VoIP Traffic

2.2.3.1 Delay

Delay or latency is the time needed to process a voice sample from input to output by the system displayed in Figure 2.10. The ITU calls this time end-to-end one-way delay[41]. The end-to-end delay is the sum of delays added by the sender, by the network transmission and by the receiver.

At the sender, delay is generated by encoding, packetization and serialization[36]. Encoding delay is a matter of the used speech codec. As discussed in section 2.2.2.1, a codec takes buffer of audio samples and compresses it to one frame. Therefore, the coding delay is dependent on the buffer size and the time needed to compress the buffer. If the codec uses a look-ahead buffer also the size of the look-ahead buffer increases coding delay[41]. The time required to fill one packet with encoded frames is called packetization delay. Serialization delay is the time needed to transmit a packet over a link. In wireless networks this is also called air link processing time [6].

When transmitting a packet over the network latency is added by queuing it in routers, serializing it and by forwarding it to the next hop. The delay from sender to receiver might not be the same as on the way back, since packets in packet-switched networks can take different paths in both directions[42].

At the receiver side delay is generated by the jitter buffer, depacketization and decoding[6]. Depackatization and decoding increase the delay by approximately the same amount as packetization and encoding do at the sender side.

[41] recommends a maximum end-to-end one-way delay of 150ms. A delay above 150ms is noticeable by the speaker and a delay above 300ms is regarded as poor connection quality by some users[41]. The implications of delay on perceived connection quality will be discussed in section 2.2.4.

2.2.3.2 Packet Loss

When packets are transmitted over the network, they can get lost or corrupted. Typical sources of packet loss are network congestion and transmission errors[6]. Sometimes, packets are dropped deliberately[36] to avoid congestion in the network. In correctly installed wired Ethernets packet loss is primarily caused by network congestion. In 802.11 networks, packet loss is also caused by channel noise and colliding transmissions. Also, in 802.11 packet loss cannot be detected by the MAC layer in all cases (hidden terminal problem), whereas the CSMA/CD mechanism in wired Ethernet detects packet loss reliably.

As described in chapter 2.2.2.3 the voice frames are transported over the network using UDP and RTP. Corrupted packets are identified by the UDP checksum. The RTP sequence number can be used to restore the correct packet order and find gaps in the packet stream. VoIP is time critical; therefore, usually no retransmission of corrupted or lost packets is initiated. Rather a technique called *packet loss concealment* (PLC) is used[42]. Here the decoder tries to fill the gaps created by packet loss. It can do that by simply replay the last received packet or by approximating the content missing packet[6].

The *packet loss ratio* is the average number of packets lost divided by the number of sent packets. A packet is also counted as lost, when it arrives at the destination but cannot be processed because of too high jitter or too high delay. The acceptable loss ratio is dependent on the codec and packet size. Therefore, the ITU[43] does not give a general advise for the maximum packet loss ratio. The ITU recommendations on packet loss are reflected in the E-Model (see chapter 2.2.4.2). By looking at the packet loss concealment mechanism described above it should be clear, that the loss of n consecutive packets has a more severe impact on speech quality than the loss of n non-consecutive packets[42]. As shown in [44] the loss ratio over a sliding window of 100 packets can vary strongly from a five packet window, as a smaller window size better corresponds with the error correction algorithm. For that reason sometimes a distinction between packet loss and bursty packet loss is made.

2.2.3.3 Jitter

Jitter is the variation of delay of consecutive packets. It is a result of different queue lengths on the network, varying paths and congestion[6, 41]. If, due to jitter, the difference of arrival times of two succeeding packets is greater than the processing time of the first packet, the decoder cannot process the second packet in time. To smoothen the jitter, all incoming packets are stored in a jitter buffer. There they are held for a certain time and afterwards released to the decoder (see Figure 2.10). A fixed jitter buffer holds back incoming packets

for a fixed time, whereas an adaptive jitter buffer adjusts the delay dynamically to the network condition[6]. A larger jitter buffer can compensate a larger jitter, but also introduces more delay.

2.2.4 Quality Metrics

The previous section covered the characteristics of VoIP traffic and how they influence connection quality. However, the question remains how the quality of a VoIP connection is defined. It can either be measured by human judgement, computational models or a combination of both. Quality metrics are important to compare the quality of two connections. This section will give a brief overview on how to obtain such indicators.

2.2.4.1 Subjective Metrics

%GoB and %PoW

A very simple method to measure voice quality is loss/noise grade model by AT&T[42]. Here volunteers rate the quality of sample calls by given categories. %GoB denotes the percentage of calls rated “good” or better and %PoW the percentage of calls rated “poor” or worse.

Mean opinion score (MOS)

The mean opinion score (MOS) is also obtained by questioning volunteers. The ITU defines how the MOS has to be measured in [45]. Test persons rate a number of voice samples on a scale from bad to excellent. The ratings are translated into numbers from 1 to 5. The average of all ratings is the MOS. A MOS of 4 is regarded to be the same quality as a normal landline phone call[6].

The MOS gives one single metric that for judging ear-to-ear voice quality. However the interpretation of the MOS is not as easy as it might seem. As stated in [42] the MOS is only a statistical sample of some subjective judgements. Therefore, a reasonable interpretation of a MOS result should also take the test environment and setting into account. Nevertheless the MOS can be very useful when comparing two systems tested in the same environment.

2.2.4.2 Objective Measures

E-Model

The E-Model is a computational model proposed by the ITU-T [46] to measure end-to-end speech quality. The model describes how to calculate the transmission rating factor *R*, which is a scalar quality metric with a maximum of 100. The *R* value can be calculated as

$$R = Ro - Is - Id - Ie - Ieeff + A$$

where

- *Ro* represents the signal-to-noise ratio,
- *Is* denotes the impairments occurring simultaneously to the signal,
- *Id* stands for the impairments caused by delay,
- *Ie* is the equipment impairment factor,
- *Ieeff* is a combination of impairments caused by low bit-rate codecs and packet loss and
- *A* represents advantages to the access of the user, that compensate the impairments.

The ITU explains in detail, how the single impairment factors can be determined. For example in [43] the *Ie* factor for a number of codecs under different packet loss assumptions are given. For instance, G.729 with voice activity detection at an packet loss ratio of 2% has an impairment factor of 19. From [43] it can be seen, that the impairment factor is highly dependent on the codec and the use of packet loss concealment.

The R-value is very useful when planning VoIP transmissions. In [32] the Telecommunications Industry Association shows which *R*-values can be achieved with different codecs, packet loss ratios and end-to-end delays. Figure 2.12 illustrates this for G.729 in comparison to G.711 (no packet loss concealment). The diagram shows that packet loss in generally is crucial to voice quality, whereas delays up to 150ms do not result in any major decrease of quality.

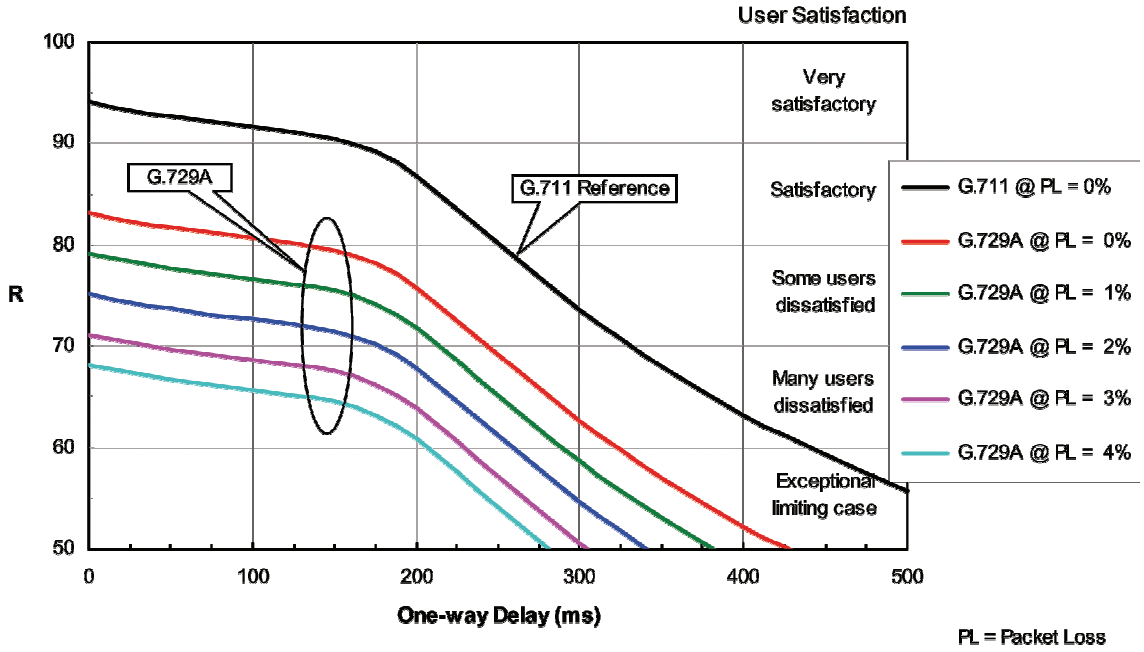

Figure 2.12 G.729A Packet loss performance, Source: [32]

Figure 2.12 also suggests that there is a direct relation between the *R*-value and the user perception. This relation between the R-value, MOS, GoB and PoW is defined in [46] and displayed in Table 2.3.

R-value (lower limit)	MOS (lower limit)	%GoB (lower limit)	%PoW (upper limit)	User satisfaction
90	4.34	97	0	Very satisfied
80	4.03	89	0	Satisfied
70	3.60	73	6	Some users dissatisfied
60	3.10	50	17	Many users dissatisfied
50	2.58	27	38	Nearly all users dissatisfied

Table 2.3 Relation between R-value, MOS, %GoB and %PoW. Source: [46]

Psychoacoustic Models

Psychoacoustic models assess the likely user perception of audio signals by objective methods. One of the first psychoacoustic models was the Perceptual Speech Quality Measure (PSQM), which was released as the ITU standard P.861. Its aim is the "estimation of the subjective quality from the objective measurement results and an analysis of the results"[47]. As stated in [42] the measurement process consists of four steps:

1. Pre-processing of the reference and degenerated signal
2. Calculation of the difference between the signals
3. Interpretation of the differences
4. Assignment of associated estimates of subjective measures of voice quality

The output of the process is the PSQM-score, which is a measure of the subjective quality degradation. The PSQM-score cannot be directly mapped into a MOS-score. The PSQM produces invalid results, when the comparison between the signals gets out of synchronisation. This shortcoming has been eliminated in the Perceptual Evaluation of Speech Quality (PESQ) method, which is published as ITU G.862 and has replaced ITU G.861.

Psychoacoustic models are mainly useful to compare the quality of codecs. They only measure the signal degeneration, but do not take effects like delay into account. Therefore a high PSQM or PESQ-score does not necessarily correspond with a good connection quality[42].

2.3 Packet Aggregation

2.3.1 Introduction

The performance of VoIP in IEEE 802.11 is surprisingly low. By looking at the bandwidth requirements of VoIP given in Table 2.2 one could assume that a single IEEE 802.11 wireless link at 2 Mbps can support dozens of concurrent VoIP calls. For example G.729 sends 20ms samples at a rate of 8 Kbps, when adding the IP/UDP/RTP overhead the sending rate is 24 Kbps. Without considering layer 1 and layer 2 overhead, naïve estimation would result in $2Mbps/(2*24Kbps) \approx 42$ calls. However, experimental results show that the actual capacity is 8 concurrent calls of a single wireless hop[48]. In multi-hop WLANs the performance degrades exponentially with the number of hops, resulting in only one supported call over a five hop string topology[48, 49]. The question, why the VoIP capacity of WLANs, especially multi-hop WLANs is so low and how it can be increased will be answered in the following sections.

2.3.2 Performance Analysis of VoIP in WLANs

Table 2.4 shows the single components of the transmissions of a G.729 frame and their channel utilization time. The base rate is 1 Mbit/s and the data rate is 2 Mbit/s. The PCLP preamble and header are sent at the base rate, the rest is sent at the data rate. The transmission of the payload and the IP/UDP/RTP header takes only about 21%, whereas the rest of the transmission time comprises of physical and MAC layer overhead.

Component	Size (bytes)	Time (µs)	Fraction (%)
DIFS	-	50	4.22%
Average channel access delay[3]	-	310	26.14%
PLCP Preamble and Header	24	192	16.19%
MAC header + CRC	34	136	11.47%
IP Header	20	80	6.75%
UDP Header	8	32	2.70%
RTP Header	12	48	4.05%
G.729 Payload	20	80	6.75%
SIFS	-	10	0.84%
PLCP Preamble and Header	24	192	16.19%
ACK	14	56	4.72%
Total		**1186**	**100%**

Table 2.4 Transmission times of a G.729 frame over a IEEE 802.11 2 Mbps link

Sending one G.729 frame over a 2 Mbit/s IEEE 802.11 link takes 1186 µs. Consequently 880 frames per second can be sent. One VoIP call requires 100 packets/second (50 per direction). Therefore the calculated capacity of a 2 Mbps link is 8.8 calls, which corresponds with the simulation results in [48].

The number of supported calls in a multi-hop string topology degrades with a factor of up to 1/*n*, where n denotes the number of hops. According to[50] the sharp decrease is a result of the following factors:

- self interference
- packet loss over multiple hops
- high protocol overhead for small VoIP packets

A solution for reducing self-interference is the use of multiple channels. [48] shows that the use of non-overlapping channels can double the VoIP capacity in the WMN. Packet loss can be reduced by new MAC schemes and error correction mechanisms. The protocol overhead can be lowered by using packet aggregation. The rest of this paper will show different approaches to implement packet aggregation for VoIP traffic.

2.3.3 Packet Aggregation Overview

Packet aggregation means to assemble one large *aggregation packet* from multiple small packets. It is called packet aggregation when it operates at IP-level and *frame aggregation* or

[3] In IEEE 802.11 CWmin is 31. Consequently the average channel access delay is 15.5 * *slot-time* = 310 µs[50]. This in only valid for non congested networks.

frame concatenation when it operates at MAC level. The opposite of aggregation is *fragmentation*, where a big packet is split up into several smaller ones.

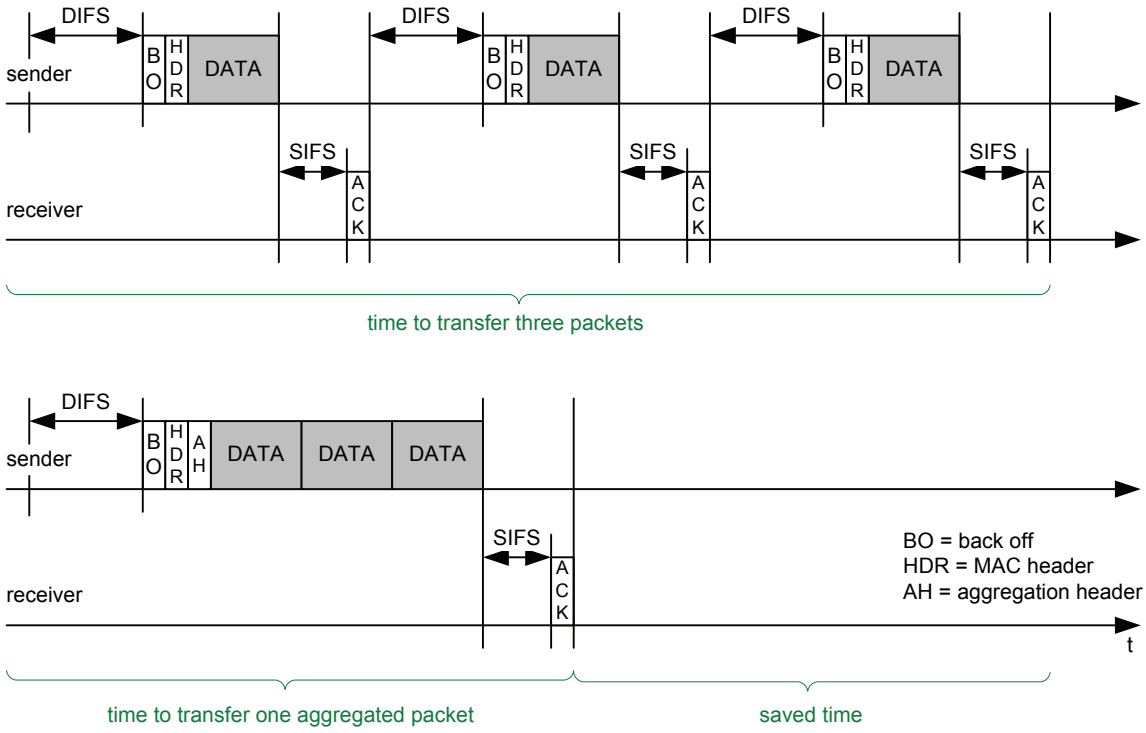

Figure 2.13 Aggregation of three packets. Source: [51]

Figure 2.13 shows the transfer of three packets with and without aggregation. Three packets are sent in one aggregation packet. The sender adds an aggregation header so that the receiver can deaggregate the packets correctly. The packet aggregation reduces the physical and MAC layer overhead and thereby saves transmission time.

If packets of the same size s_p (in bytes) are sent at a channel rate r (in Mbps), the saved time t_s when aggregating n packets can be calculated as:

$$t_s = n * (t_o + \frac{8 * s_p}{r}) - (t_o + \frac{8 * (AH + n * s_p)}{r}) \quad (1)$$

Where AH denotes the size of the aggregation header in bytes and t_o the channel time for DIFS, two times the physical layer overhead, the MAC header, SIFS, ACK and the average channel access delay (BO). When simplifying equation 1 to

$$t_s = t_o * (n-1) - \frac{8 * AH}{r} \quad (2)$$

it can be noticed the benefit t_s gained from aggregation increases with the number of packets.

Although the equation implies "the more aggregation the better" the implementation of an aggregation mechanism for VoIP requires additional considerations. The following questions need to be considered:

- How to design an aggregation algorithm that supports QoS? The aggregation algorithm should ensure that the VoIP traffic fulfils certain packet loss, delay and jitter constraints.
- How to choose the optimum *n* (i.e. packet size)? The size of the aggregation packet plays an important role for the performance. The optimum size maximizes the capacity of the network.
- How can redundant header information be compressed? Aggregation creates redundant information in the IP/UDP/RTP headers, which can be compressed best using compression schemes designed for packet aggregation.
- On which OSI layer should the aggregation be performed? Either audio frames, MAC frames or IP packets can be aggregated. All methods have benefits and drawbacks.

Answers will be given in sections 2.3.4, 2.3.5, 2.3.6 and 2.3.7.

2.3.4 Aggregation Algorithms for VoIP Traffic

One basic design decision for an aggregation algorithm in a WMN is the placement of (de)aggregation capabilities. Two different approaches – *end-to-end aggregation* and *hop-by-hop aggregation* – are presented in [52], [51] and [50]. In end-to-end aggregation the sender or ingress node aggregates the packets and the receiver or egress node deaggregates the packets. Intermediate nodes just forward the aggregated packets. Therefore the aggregation is transparent to the backbone network and easy to implement. In contrast, every (backhaul) node de-aggregates and aggregates when using the hop-by-hop aggregation scheme. Thereby an efficient inter-flow aggregation is possible.

A node needs to collect packets in a queue, before it can aggregate them into a larger packet. A very simple approach to do this is *forced-delay aggregation*. Here each incoming packet is marked with a timestamp. Each packet is delayed for a *maximum delay* period. After this period packets with same next hop are aggregated. The maximum number of packets in an aggregation packet is limited by the *Maximum Transmission Unit* (MTU) size, which is about 1500 bytes for wired Ethernet and 2300 bytes for IEEE 802.11[10]. If a packet cannot be aggregated with others, it is sent as it is after the maximum delay period expires. The right choice of the maximum delay period is important. Higher delays yield a higher aggregation

rate, but also a higher end-to-end delay. In addition, the maximum delay period directly influences packet size.

When using a forced-delay end-to-end aggregation the delay is introduced only at the ingress node. In a hop-by-hop scheme however, every node adds additional delay. However, it is not always necessary to artificially delay packets. Due to the media access delay packets are delayed anyway. The *accretion aggregation* algorithm[51] makes use of this fact. Forced delay is only added at the ingress node to collect packets of the same flow. The intermediate nodes do not further add forced delay but use the natural media access delay. [51] shows, that this algorithm outperforms the forced-delay hop-by-hop and end-to-end approach.

2.3.5 Adaptive Aggregation Algorithms

In section 2.3.3 it was argued that a higher aggregation ratio (i.e. the ratio of aggregated packets) results in a lower overhead and theoretically a better performance. However, a higher aggregation ratio does not necessarily lead to a better throughput. This is because larger packets are more likely to get corrupted by bit error or interference than smaller packets. Adaptive aggregation schemes that adjust their maximum frame size to the channel quality or network contention will be presented in the following paragraphs.

2.3.5.1 Frame Aggregation and Optimal Frame Size Adaptation for IEEE 802.11n WLANs

For a given local traffic condition and channel quality an optimum frame size can be found[53-55]. In [55] a model for calculating this size in single-hop WLANs is presented. Figure 2.14 displays the relation between packet size, channel quality, the contention level and network throughput. For different numbers of 100 byte MAC Service Data Units (MSDUs – i.e. MAC frames) the network throughput in a 54 Mbit/s 802.11 network is plotted. The diagram shows that the optimum frame size L^* is dependent on the bit error rate (BER). L^* is small on a link with a high BER and large on a link with a low BER. In the paper a model for calculating the successful transmission probability of a frame of a certain length is proposed. The level of network contention – here expressed by the number of stations n in Figure 2.14 – does only have a minor influence in this model.

Based on these observations two different MAC frame aggregation schemes are proposed in [55]. The adaptive aggregation shows higher performance than fixed frames sizes. However, the paper does not include details on how the frames are delayed. Also the model was developed and verified for single hop WLANs only. Since WMNs do suffer from self-interference whereas it is not a big problem for legacy IEEE 802.11 LANs, the results might

not be applicable for WMNs. Also, the model includes collision probabilities in its calculations. Due to the hidden terminal problem those probabilities are different in multi-hop WLANs than in single-hop WLANs.

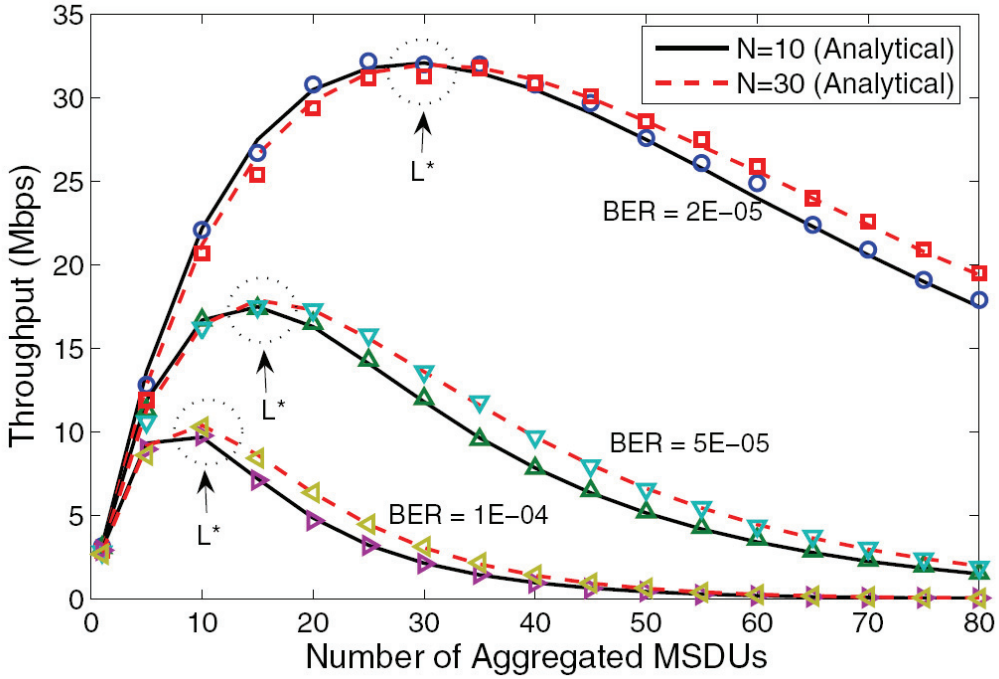

Figure 2.14 Network throughput with different packet sizes, Source: [55]

2.3.5.2 IP based Adaptive Packet Concatenation (IPAC)

The "IP-based Adaptive Packet Concatenation Multi-hop Wireless Networks"-algorithm[56] is an end-to-end aggregation scheme. The basic idea behind this method is to select the packet size based on the route quality. It uses the Weighted Cumulative Expected Transmission Time (WCETT) routing metric to determine the Maximum Concatenation Interval (MCI) and the Maximum Concatenation Size (MCS). Packets at the ingress node are delayed by MCI. The aggregated packet size cannot exceed the MCS.

The authors of [56] determined the optimum values for MCI and MSC at a given WCETT in simulations. The algorithm itself uses a quadratic function that calculates this mapping approximatively. IPAC performs especially well in high loaded networks, where the throughput can be more than doubled. The end-to-end delay is increased though.

2.3.5.3 Rate Adaptive Framing

In [52] a method to adapt the frame size dynamically to the channel quality and network contention is presented. In *Rate Adaptive Framing,* the receiver calculates the optimum frame size and channel rate based on the assumption that wireless interference is predictable on the

short term. It chooses the frame sizes and channel rate so that collisions are unlikely. The receiver informs the sender about these parameters in the ACK packet.

The paper shows that for higher offered load the optimum frame size increases up to a dropping point. There it is beneficial to reduce the channel rate and packet size to minimize the interference.

2.3.6 Aggregation in the OSI model

Beside the question "How to aggregate?" the question "What to aggregate?" needs to be answered. The schemes in sections 2.3.4 and 2.3.5 either aggregate MAC-frames or IP-packets. A third option is to aggregate audio frames produced by the codec. For example G.729 produces 10 ms frames, but two consecutive frames are sent in one packet[33].

	Advantages	**Disadvantages**
MAC Layer	• Natural MAC delay can be used • Frame fragmentation and aggregation in one system • Aggregation can adapt to network situation • Will be part of IEEE 802.11n • Can be combined with new MAC schemes	• Only hop-by-hop, but could be end-to-end with 802.11s • Needs changes in MAC layer – which is part of the hardware firmware • Not compatible with IEEE 802.11a/b/g
Network Layer	• Efficient Header Compression • Aggregation can adapt to network situation • Transparent to application • End-to-end and hop-by-hop is possible • Can be transparent to intermediate nodes	• MAPs are usually not layer 3 devices • Requires changes to IP stack
Application Layer	• Transparent to lower layers • Sometimes part of the codec • Creates additional delay • Reduces overhead very efficiently	• No intra-flow aggregation possible • Only end-to-end • Application needs to be changed • Very sensitive to bursty packet loss

Table 2.5 Comparison of aggregation on different layers

Table 2.5 shows the advantages and disadvantages of these three approaches. It should be noticed that they are not mutual exclusive. For example the codec can aggregate two audio frames in one RTP packet, which are then further aggregated on the network layer.

2.3.7 Header Compression

A technique complementary to packet aggregation is header compression. As pointed out in section 2.2.2.3 header compression exploits the fact that most header fields only change little or stay static during a VoIP session. Although protocols like ROHC[40] work efficiently also over wireless links better solutions can be found in connection with packet aggregation.

A scheme called *Zero-Length Header Compression* (ZLH) is proposed in [51]. When packets within one flow are aggregated only the header of the first packet is needed. The other headers can be restored out of the first header. This dramatically reduces the overhead and doubles the throughput.

ZLH can be used in combination with ROHC. Hereby, the header of the first packet is compressed using RHOC, which reduces the overhead from 40 to 14 bytes. The headers of the following packets are compressed with ZLH, which totally removes the overhead.

2.3.8 Summary

Packet aggregation is a suitable method to improve the VoIP capacity of a WLAN. If the aggregation algorithm is designed in a right manner, the performance can be more than doubled while still maintaining the QoS criteria packet loss ratio, delay and jitter. Different aggregation schemes have been proposed in the literature. A comparison of these schemes is given in Table 2.6.

Aggregation Scheme	**Suitable for WMNs**	**Advantages**	**Disadvantages**
Forced Delay	Yes	Easy to implement	Adds delay, even when not necessary
Accretion	Yes	Uses natural delay	Does not consider current network condition
IPAC	Yes	Adaptive to network condition	Only end-to-end aggregation
Rate Adaptive Framing	Yes	Adaptive to network condition	Requires changes in MAC protocol
Optimal Frame Size Adaptation for IEEE 802.11n WLANs	No		Single-hop only

Table 2.6 Comparison of aggregation schemes

For WMNs it is favourable to use an adaptive hop-by-hop aggregation scheme. Thus, aggregation can be adapted to the local network condition. In the next chapter a novel

adaptive aggregation scheme for WMNs will be presented, which adapts the maximum packet size to a value, which determined by the receiver upon the link quality.

3 Design of a new Packet Aggregation Scheme

This section describes in detail what the problems and challenges of packet aggregation combined with VoIP are. Especially the role of the packet size will be discussed. Two approaches for determining the packet size and one for designing an aggregation mechanism are outlined.

3.1 Problem Definition

In the previous chapter the concept of aggregation and challenges of applying packet aggregation to VoIP were discussed. The presented solutions increase the VoIP capacity of WMNs. However they are either not adaptive to the current network condition or they are not flexible since they use end-to-end aggregation or they are not designed for multi-hop networks. Therefore, it is desirable to design and implement a new packet aggregation mechanism that meets the following criteria:

3.1.1 Design Criteria

3.1.1.1 Adaptive Aggregation

From [56] and [54] it can be inferred that the packet size is a very important factor for efficient aggregation. This value needs to be determined adaptively based on the current link quality and network load. The trade-off between large packets (loss due to bit errors) and small packets (loss due to contention) will be explained in section 3.2.2.

3.1.1.2 Hop-by-Hop Aggregation

In many scenarios hop-by-hop aggregation will achieve a better aggregation ratio than end-to-end aggregation, since more packets can possibly be aggregated. In addition, end-to-end aggregation is not efficient if a route consists of links with different quality. As it will be discussed later, the maximum packet size is limited by the link quality. If end-to-end aggregation is used, the maximum packet size of a whole route is limited by the weakest link. This is also a major shortcoming of the IPAC approach. In contrast, hop-by-hop aggregation can use larger packets on good links and smaller packets on bad links.

3.1.1.3 Enhance Performance of Voice over IP Traffic

The aggregation should enhance the performance of VoIP connections in a WMN by reducing the packet loss ratio and the end-to-end delay. As discussed in Chapter 2 a network needs to fulfil QoS criteria like packet loss ratio, maximum end-to-end delay and maximum jitter to support VoIP.

Performance improvement of VoIP in WMNs can aim for different targets:

1. Improving the quality of a single VoIP call with background traffic in place or
2. maximizing the quality of a given number (>1) of VoIP calls with no background traffic or
3. maximizing the number of supported connections while staying within the QoS parameter constraints or
4. a combination of the later ones. Maximizing VoIP capacity while having non-VoIP traffic in place.

Although all three targets sound quite similar the ways to achieve them are different. The first goal can typically be reached by *traffic prioritization* and *resource reservation protocols*. Packet aggregation can be an optional mechanism in this case. Aggregation is one method to achieve the second and the third target. The fourth problem can be solved with traffic shaping or resource reservation and packet aggregation.

The aggregation scheme should primarily address the third goal, which can be split up into two subsequent goals:

- Minimizing the packet loss ratio
- Minimizing delay and jitter

Minimizing the packet loss ratio is the primary goal, staying within delay and jitter constraints are only considered subsequent goals here, since packet loss is the more critical factor to the speech quality (see Figure 2.12 and [57]). Because of the 802.11 MAC retransmission mechanism minimizing packet loss has also beneficial effects on delay and jitter.

3.1.1.4 Aggregation of IP Packets

As shown in Table 2.5 aggregation can be performed in different positions on the network stack. The aggregation of IP packets is more flexible than MAC frame aggregation and can be extended with header compression and network coding.

3.1.2 Research Questions

The design criteria lead to the following research questions:

- How can an aggregation algorithm by implemented? Section 3.2.1 will give an answer by sketching the important components of an aggregation mechanism.
- How can IP packets be aggregated? Section 3.2.1.2 will describe an encapsulation concept.
- What packets to aggregate? Section 3.2.2 illustrates how the packets for aggregation can be selected.
- What is the optimum packet size? In section 3.2.2 the implications of packet size on packet loss ratio will be shown.
- How can the optimum/maximum packet size be determined? In sections 3.2.2 and 3.2.3 methods for obtaining the optimum/maximum packet size will be developed.
- Who should determine the size? The sender or the receiver? Section 3.2.2 gives reasons why the receiver should do determine the size.
- How can the packet size information be exchanged? In section 3.2.3 two novel protocols will be presented.

3.2 Suggested Solutions

3.2.1 Aggregation of IP Packets

The concept of packet aggregation is:

- Collect packets which pass a common hop (aggregation target)
- Aggregate packets together in an aggregation packet. Send this packet to the aggregation target
- Deaggregate the aggregated packets at aggregation target

3.2.1.1 Collect packets

A node first needs to collect packets in order to aggregate them. The best place to do that is the outbound queue in the MAC layer. At this point all necessary information about a packet, such as the IP and MAC-addresses of the next hop, is available. This queue stores packets, which are scheduled to be sent. Ideally, packets are spread over different virtual queues by their class of service. Thereby VoIP traffic can be prioritized over other traffic. If the traffic load is high enough, packets queue up due to the natural MAC delay. However, in low traffic settings some artificial delay needs to be introduced by not sending packets, even if the MAC layer is idle.

3.2.1.2 Aggregate packets into a large packet

Packets in the outbound queue are aggregated upon their common aggregation target. A new IP packet – the *aggregation packet* - is created. The *aggregated packets* P_0-P_n are the payload of the aggregation packet (see Figure 3.1). The IP_META IP-header is added by the aggregation algorithm. It represents the aggregation header from Figure 2.13. Because of the additional IP-header the overhead generated by the aggregation is 20 bytes. To identify an aggregation packet, the *protocol* field of the IP-header is set to IP_META (the numerical value). When hop-by-hop aggregation is used, the aggregation target is the next hop of the aggregated packets. For end-to-end aggregation it is the end of the aggregation tunnel.

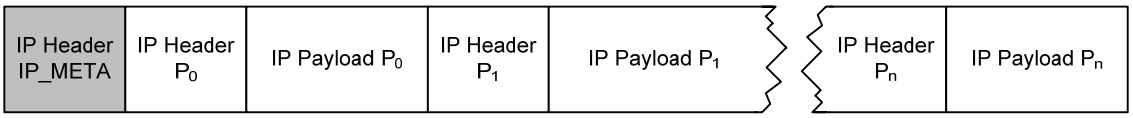

Figure 3.1 Encapsulation concept

3.2.1.3 Deaggregate at the aggregation target

When the aggregation target receives an aggregation packet, it can identify it by looking at the *protocol* field. It can access the single aggregated packets by reading the *length* field of the first packet, calculate the offset of the second and so on. After extracting all aggregated packets it can pass them further to the upper OSI-layers and discard the aggregation packet's IP-header.

3.2.2 Determination of the Optimum Packet Size

For a given channel quality, contention level and traffic injection rate (rate at which the nodes generate traffic) different packet sizes produce different packet loss ratios. In order to minimize the packet loss it is desirable to determine the optimum frame size[4], for which the packet loss ratio is minimal. However, it is not obvious how that can be done. In the literature [53, 55, 56, 58] different methods for obtaining the optimum frame sizes are proposed. For determining the frame size, which minimizes the packet loss, it is important to investigate the reasons for packet loss. For example, those are:

- Channel Bit Errors
- Collisions

[4] The frame size is the sum of packet size plus the MAC header size. If the frame size is known the packet size can easily be calculated.

- Queue Overflows
- Route errors

Since route errors cannot be influenced directly by the packet size, only the first three causes of packet loss will be discussed.

Packet Loss due to Bit Errors

One cause of packet loss is loss due to bit errors in the transmitted frame. Bit errors occur when the received signal cannot be decoded properly. The extent of bit errors is dependent on the signal-to-noise ratio (SNR) of the received signal, the modulation scheme, the coding scheme and the data rate[59, 60]. The SNR is defined as[61]:

$$SNR = 10\log_{10}(\frac{P_{signal}}{P_{noise}})$$

- where P_{signal} is the strength of the signal
- and P_{noise} is the strength of noise produced by thermal noise and interference.

If the other factors are considered to be fixed (which is the case – for example - in an IEEE 802.11b WLAN at a fixed data rate), the BER is only dependent on the SNR. The frame error rate (FER) depends on the bit error rate. For a given BER the FER increases exponentially with the frame size. A simple analytical approach for calculating the FER is[62]:

$$FER = 1 - (1 - ber(SNR, R_{pre}))^{L_{pre}} * (1 - ber(SNR, R_{plcp}))^{L_{plcp}} * (1 - ber(SNR, R_{mac}))^{8*L_{mac}}$$

Where

- R_{pre} is the transmission rate of preamble,
- R_{plcp} is transmission rate of PLCP header,
- R_{mac} is transmission rate of MAC frame,
- L_{pre} is length of preamble in bits,
- L_{plcp} is the length of PLCP header in bits and
- L_{mac} is the length of MAC frame in bytes.

If the length of the preamble and header and the transmission rates are considered to be constant, the FER is a function of the SNR and the frame length.

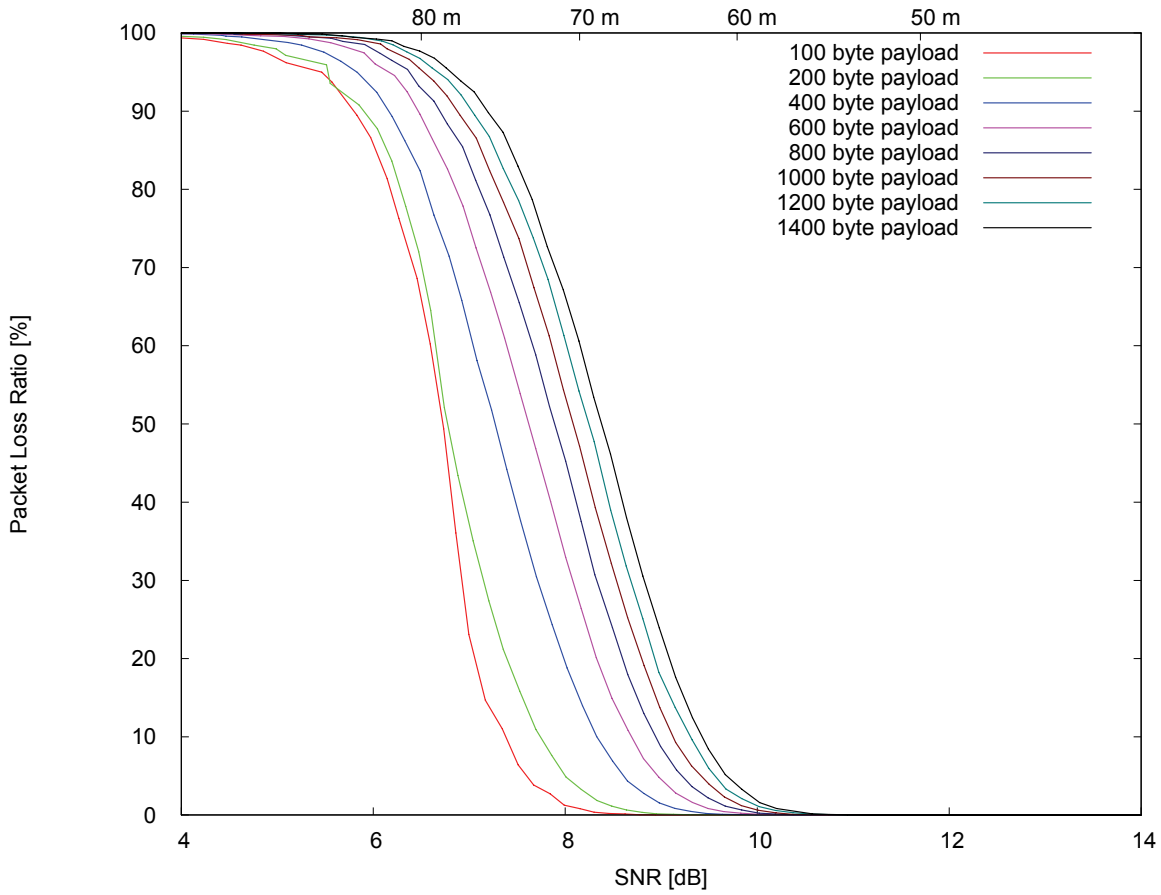

Figure 3.2 Packet Loss Ratio as a function of the SNR and packet size

Figure 3.2 shows the correlation between packet size (IP-payload size), SNR and packet loss ratio due to frame errors. It was obtained by simulating the transmission of packets at 64kBit/s between two wireless nodes. The SNR was varied by changing the distance between the nodes after each simulation run. A SNR of 12 dB is obtained at a node distance of about 50 m, a SNR of 10 dB at approximately 60 m etc. For each node distance packets of different size were sent and the packet loss ratio was calculated. Table 5.1 lists the other simulation parameters.

The figure clearly shows that the packet loss due to frame errors is lower for small packets. Especially for a low SNR the difference between packet loss with small and with large packets is great. For example at 70 meters distance packets with 200 byte payload have a loss ratio of almost zero, while approximately 30% of the 1400 byte payload packets are lost.

Packet Loss due to Collisions

If the network load and the number of contending nodes rises, the probability of collisions on the MAC layer rises. Larger packets have a better payload/overhead ratio. Therefore they reduce the channel utilization and consequently the probability of collisions. However, too

large packets will cause packet loss due to bit errors, MAC layer retransmissions and congest the channel even more.

Packet Loss due to Queue Overflows

If a node tries to send more data than the MAC layer and channel speed can handle, the packets will queue up in the internal packet queue. If the packet rate is greater than the maximum possible MAC service rate this will eventually lead to a queue overflow. New packets cannot be stored in the queue anymore and are dropped. Again, larger packets will reduce overhead and therefore increase the MAC service rate.

Conclusion

Determining the packet size which minimizes the packet loss ratio is a difficult problem. In multi-hop wireless networks packet loss is caused by various reasons. Sometimes larger packets decrease the packet loss ratio, sometimes smaller packets. Therefore the optimum packet size can only be found if all causes are known. This leads to the following conclusions:

- One or several metrics need to be obtained, from which the cause and amount of packet loss can be deduced
- A mapping between those metrics and the optimum packet size needs to be created
- Since some metrics are only available at the receiver, there needs to be an exchange mechanism for the metrics or the packet size information.

3.2.3 Packet Size Selection Protocols

Subsequently three proposals for a distributed calculation of the packet size and its exchange are presented.

3.2.3.1 ETX-based Packet Size Adaptation

One idea is to calculate the optimum packet size as a function of the *Expected Transmission Count* (ETX, see section 2.1.4.2). Since the ETX comprises all causes of packet loss it seems to be a good metric for determining the optimum packet size. The packet size S should be a function of the ETX:

$$S = f(ETX)$$

In order to construct f it needs to be investigated, how S correlates with the ETX. That such a correlation exists at all, can be inferred from the IPAC-paper[56]. IPAC utilizes a correlation between the *Weighted Cumulative Expected Transmission Time* (WCETT) and the packet size that minimizes packet loss ratio. In a single channel network the WCETT is defined as[25]:

$$WCETT = \sum_{i=1}^{n} ETT_i$$

where *n* denotes the number of hops. The ETT is $ETX * \frac{S}{B}$, where *S* is the packet size and B the bandwidth. If *S* and *B* are considered to be fixed, the ETX is a multiple of the ETT. For the special case of n=1 (that means the WCETT for one hop), the ETX is also a multiple of the WCETT. As a consequence - if there is a correlation between WCETT and the optimum packet size - this relation should also exist between ETX and the packet size.

Verification

To verify this assumption a number of simulations were performed. The nodes were placed in a string topology as shown in Figure 3.3. Between Node 0 and Node 3 background traffic of a constant rate and packet size was generated. Between Node 1 and Node 2 a stream with 64 kBit/s was created (same packet size as background traffic). For each simulation run the background traffic rate (0 to 1500 kBit/s), the packet size (100 to 1000 bytes) and the distance d_1 (40 to 90 m) was varied. d_0 and d_2 was 50 in all simulations.

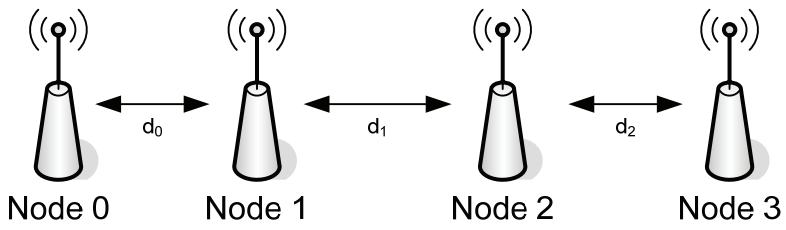

Figure 3.3 Simulation Topology for determining ETX/Size correlation

For each run the packet loss ratio of the stream between Node 1 and Node 2 and the average ETX was calculated. Afterwards the simulation outcomes with a similar ETX (interval of 0.05) were clustered. For each cluster the packet size with the lowest packet loss ratio was determined. If several packet sizes yielded the same packet loss ratio, the smaller size was selected.

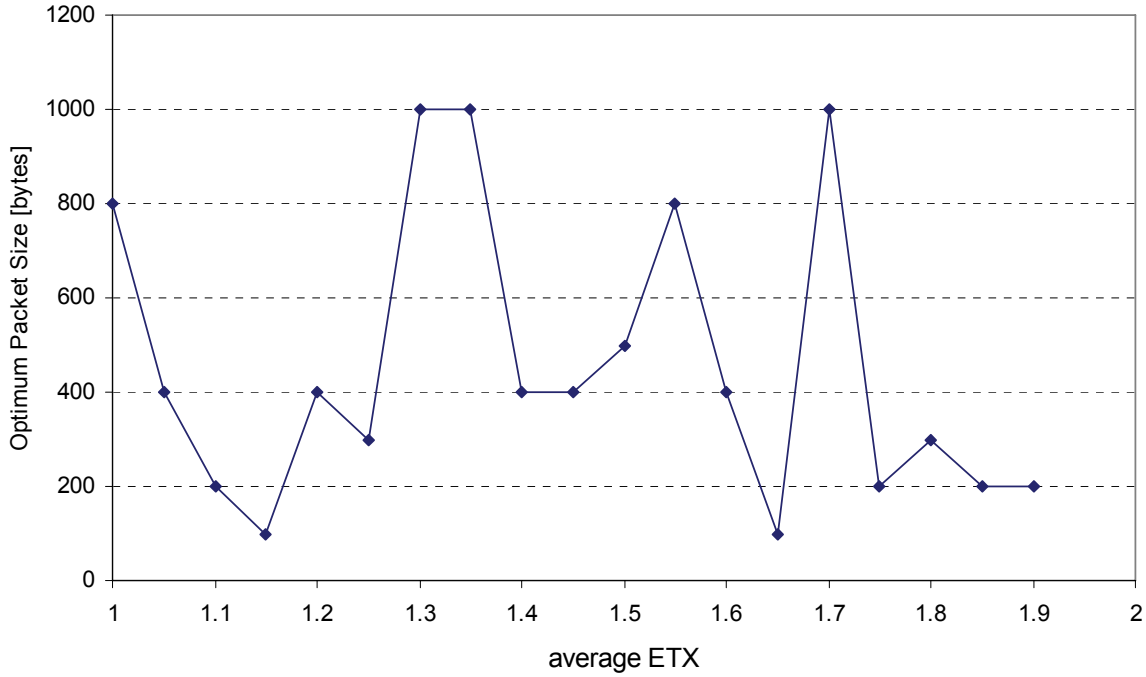

Figure 3.4 Correlation between ETX and optimum packet size

Figure 3.4 shows the correlation or the non-correlation between the average ETX and the optimum packet size. The diagram indicates that there is no correlation between the optimum packet size and the ETX, if the node distance and the network load is variable.

Conclusion

The previous assumption, that the optimum packet size and the ETX should correlate, could not be verified in the simulation. One reason can be that the size of the ETX probe packets is rather small (105 bytes including the MAC header), compared to the size of the payload packets. With a bad channel (low SNR) the probability of losing a small packet is lower than losing a large packet. So the ETX is not representative for larger packets in bad channels. [23] confirms this shortcoming of the ETX and proposes a scheme to interpolate the ETX for any packet size, by using probe packets of two different sizes. However, also the interpolated ETX did not correlate with the optimum packet size.

A different problem with the ETX based packet size selection is that the ETX does not give any information about the reason of packet loss. For example, packets can get lost due to bit errors or channel contention. Both factors can cause a high ETX. In the first case smaller packet sizes are beneficial, in the second case larger packet sizes. Therefore the ETX alone does not seem to be an appropriate metric for determining the optimum packet size. The

question why the IPAC paper[56] presents a correlation between the WCETT and the packet size must remain unanswered here.

3.2.3.2 Simple Packet Size Selection Protocol

Because the ETX-mapping did not give the expected results, a new protocol – the Simple Packet Size Selection Protocol – was developed for this paper. It enables a sender to determine the maximum packet size by information gathered at the receiver. This idea is advantageous, since contention and the channel quality should be measured at the receiver. Collision-Aware Rate Adaptation[63] also measures those parameters at the receiver and communicates the desired transmission rate to the sender. The *Rate Adaptive Framing* scheme[54] utilizes the same concept to adapt the sending rate and the frame size.

The idea of the *Simple Packet Size Selection Protocol* is that

- the receiving node constantly measures the SNR of incoming packets,
- calculates the maximum tolerable packet size based on the current SNR and
- transmits the calculated value to the sender.

In detail the protocol works as described following and as illustrated in Figure 3.5:

1. Measure the SNR of incoming packets

For each one-hop neighbour a node measures the SNR of every incoming packet.

2. Calculate smoothed SNR

To eliminate jumps in the SNR caused by short time fading, the current smoothed SNR for a link is calculated as the exponential moving average[64] of the previous smoothed SNR and the current measurement:

$$SNR_{current} = SNR_{previous} + \alpha * (SNR_{measured} - SNR_{previous})$$

where $SNR_{previous}$ is the calculated SNR before receiving the current packet, $SNR_{measured}$ is the measured SNR of the incoming packet and α is a smoothing factor. α determines the speed at which older measurements are dampened[64]. When it is close to 1, dampening is quick and when it is close to 0, dampening is slow. Since the link conditions in a static WMN should be quite stable over time, a low α - e.g. 0.2 - can be used to have a good smoothening of short time fading. For every link the $SNR_{current}$ value is stored in the routing table.

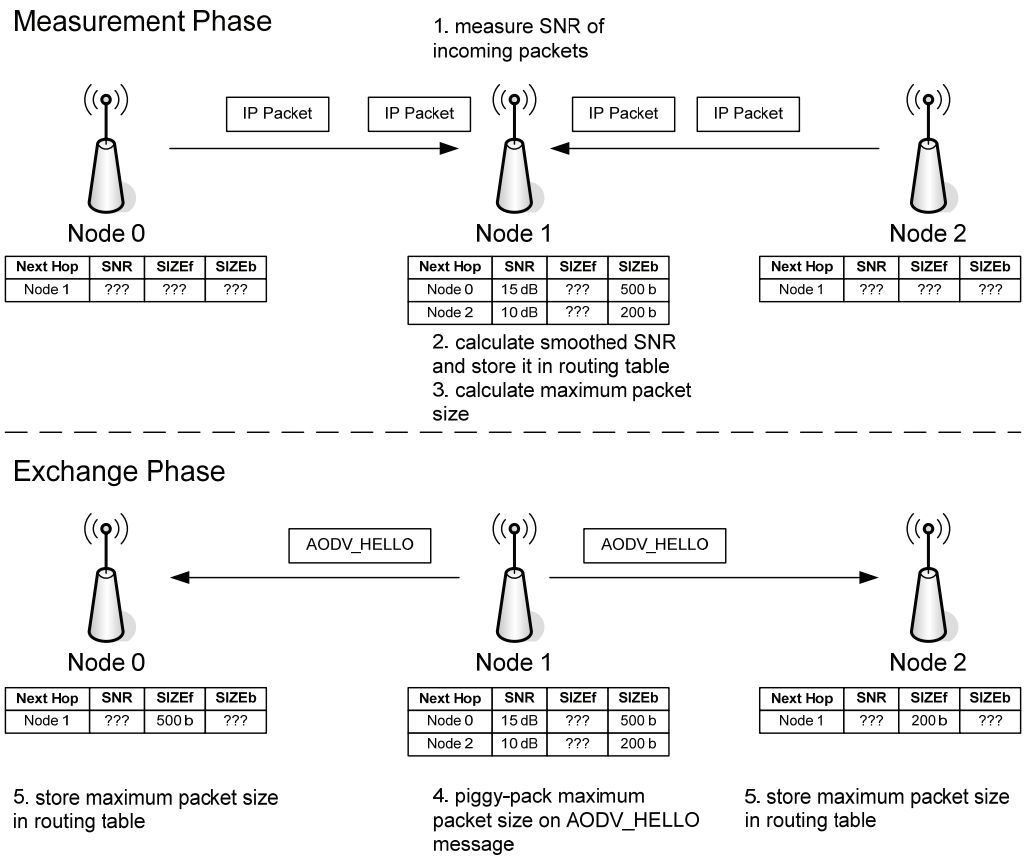

Figure 3.5 Simple packet size selection protocol

3. Calculate the maximum packet size

Figure 2.12 shows that VoIP allows a certain packet loss without major quality impairment. The overall packet loss ratio constitutes of drops due to frame errors, drops due to contention (collisions, queue overrun etc.) and routing errors. The Simple Packet Size Selection Protocol aims to reduce drops due to frame errors.

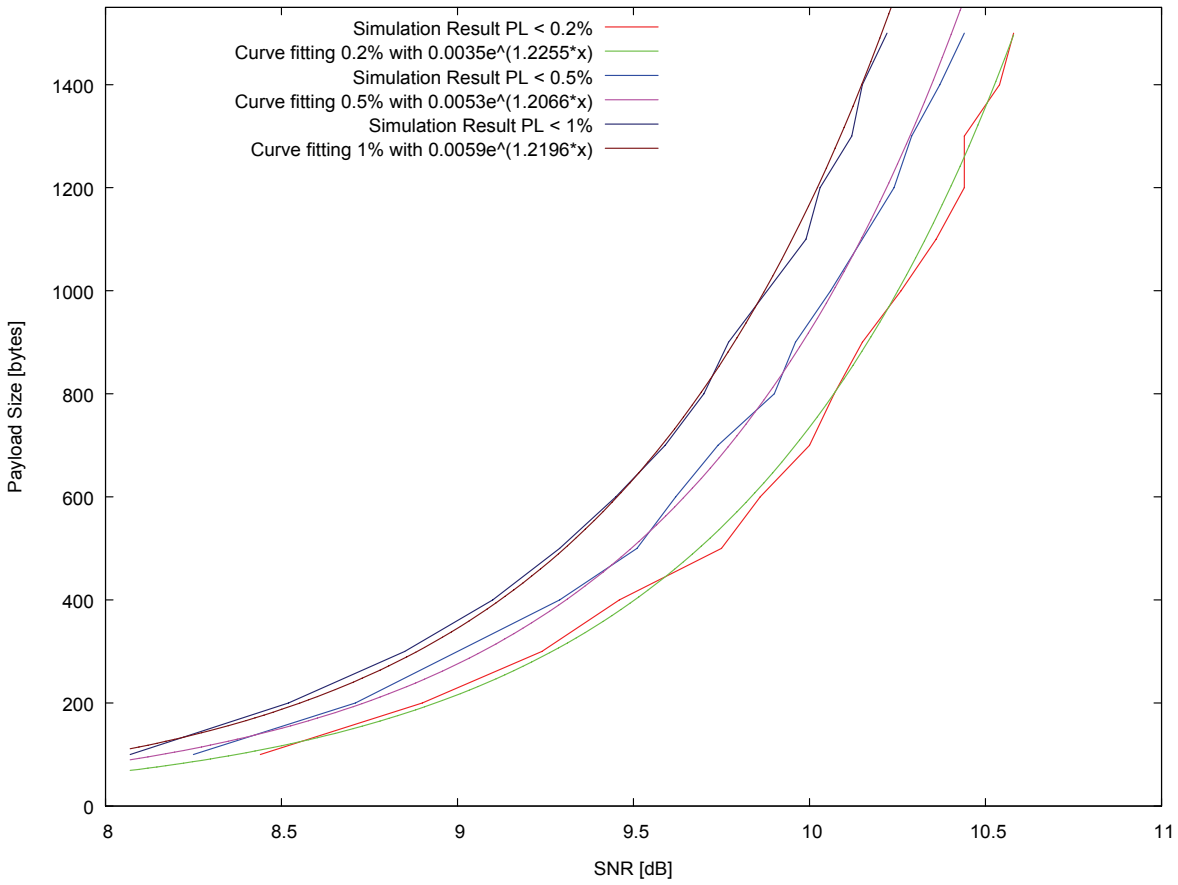

Figure 3.6 Maximum packet size with a loss ratio < 0.2%, 0.5% and 1%

The protocol assumes, that for each hop the packet loss due to frame errors (FER) should not exceed 0.2%, which would result in about 1% packet loss for five hops. Figure 3.6 shows the maximum packet sizes, which yield less than 0.2, 0.5 and 1% packet loss due to FER. It was obtained by measuring the drops due to FER for different packet sizes and SNR (by changing node distance) values. For a given packet size the lowest SNR at which the packet loss ratio is less than 0.2% (0.5 and 1% respectively) is plotted. For 0.2% the function is nearly exponential and can be approximated with

$$\text{MAX\_SIZE}(SNR) = 0.0035e^{1.2255*SNR}$$

Each node calculates this function for every neighbouring link.

4. Transmit the packet size to the sender

In the next step the calculated packet size needs to be transmitted to the sender. This can be done by piggy-packing the information on periodical exchange of route information. If no routing protocol is in place, an option is to modify the RTS/CTS packets. It is also possible to a client/server architecture, in which the sender requests the packet size from the receiver.

In the implementation in Chapter 4 the information is exchanged via AODV HELLO-messages. Nodes periodically broadcast AODV HELLO-messages for route maintenance. On those messages the node piggy-packs host-address/packet size (SIZEb) pairs.

5. Store Maximum packet size in routing table

When a node receives an AODV HELLO-message it reads the host-address/packet size pair with its own host address. It stores the packet size in the corresponding entry (SIZEf) of its routing table.

3.2.3.3 Enhanced Packet Size Selection Protocol

The Simple Packet Size Selection Protocol only considers the SNR when calculating the packet sizes. However, also the level of contention is an important factor for determining the optimum packet size. Hence it is desirable to distinguish between packet loss due to FER and packet loss due to contention – a concept which is realised by the newly developed *Enhanced Packet Size Selection Protocol* (Figure 3.7). The idea of it is

- to monitor SNR and to measure delivery rate of periodical probe packets
- to determine the current network load and link quality at the receiver
- to use a pre-calculated table for looking up the optimum packet size for the current network state
- to pass optimum packet size to the sender via routing messages

In detail, the protocol works as follows:

1. Measure SNR and delivery rate of probe packets

For each one-hop neighbour a node measures the SNR of every incoming packet and counts the number of received AODV HELLO messages.

2. Calculate smoothed SNR

Using exponential moving average a smoothed SNR is calculated for every link and stored in the routing table.

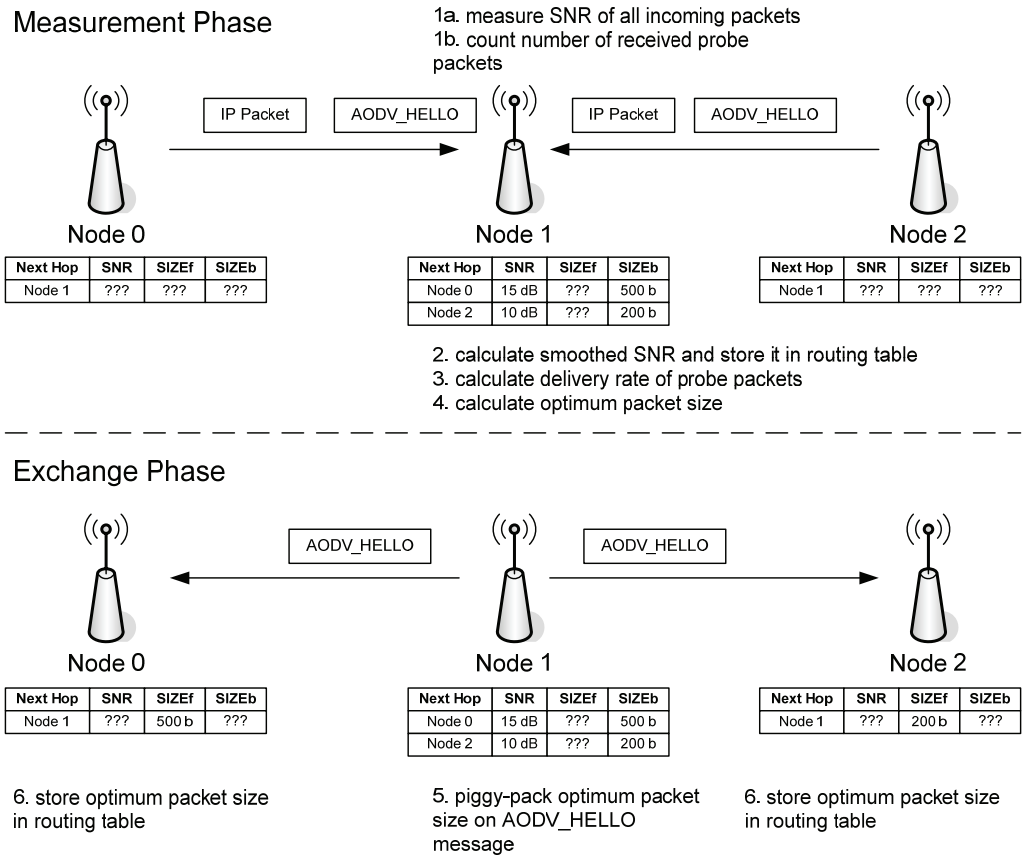

Figure 3.7 Enhanced Packet Size Selection Protocol

3. Calculate delivery rate of probe packets

Every node broadcasts one AODV HELLO message per second. So a node knows, that it should receive 10 AODV HELLO messages in 10 seconds from each of its direct neighbours. By counting the actual received ADOV HELLO messages the node calculates the delivery ratio of AODV HELLO messages for every link. Since the AODV HELLO messages are broadcasted the delivery ratio is not tampered by MAC layer retransmissions.

4. Calculate optimum packet size

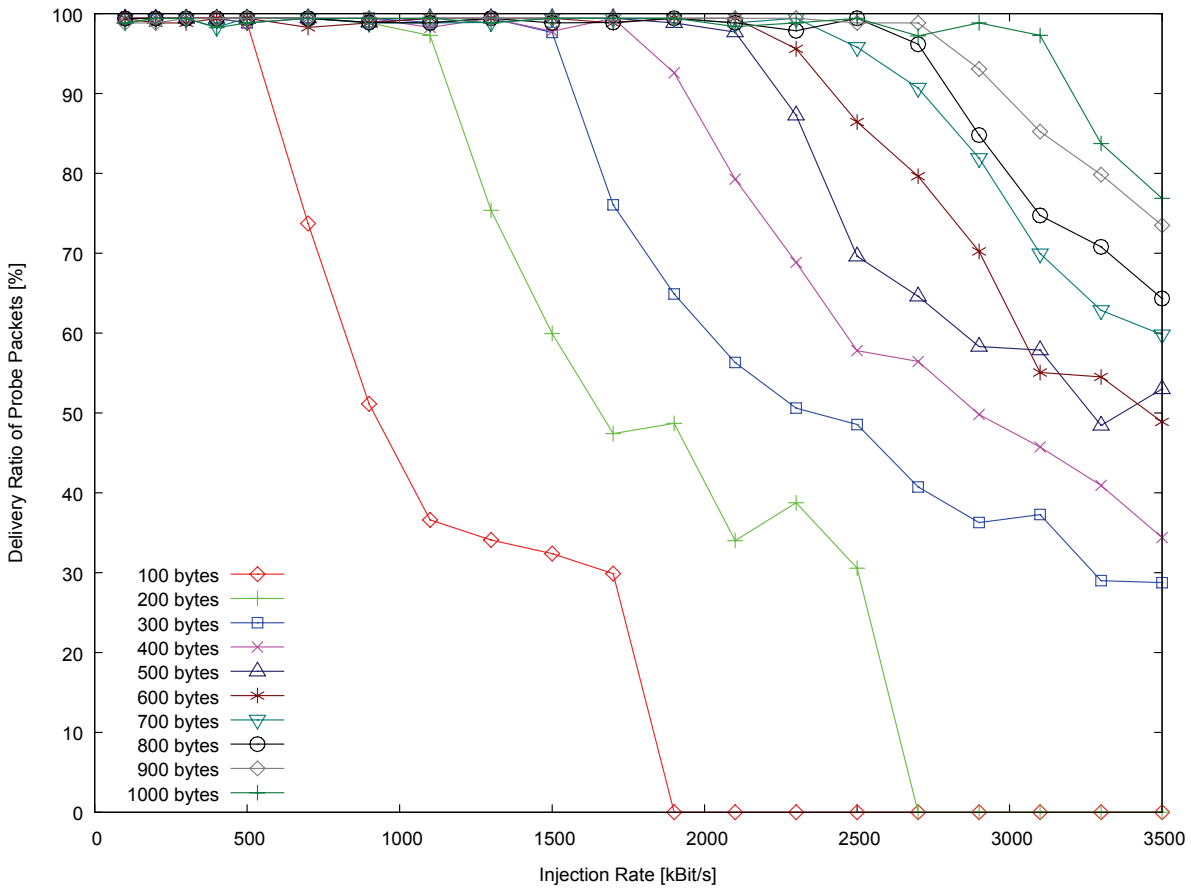

Figure 3.8 Delivery rate of probes at 10m node distance, SNR = 27 dB

From steps 1-3 a node knows the SNR and the delivery ratio of the AODV HELLO messages. It also knows the current packet size it uses. Using that information the node can roughly estimate the current network load. It does that by looking up a delivery rate/injection rate/SNR/packet size table. This table needs to be created once, by measuring the delivery rate with different traffic injection rates, packet sizes and node distances (i.e. SNR values). Figure 3.8 shows a plot of such a table for a SNR of 27 dB. On the x-axis is the network load, on the y-axis delivery rate of probe packets. It was obtained by sending constant bit rate traffic over a string topology of 4 nodes and measuring the delivery ratio for packet sizes of 100-1000 bytes and injection rates of 64-3500 kBit/s at a node distance of 10 meters. For a different topology the table would be different. E.g. a higher node density would result in more loss due to collisions at a lower injection rate.

If the size of packets in the network is 200 bytes (e.g. as a result of aggregation and fragmentation) and the delivery ratio is 70% the node can therefore deduce that the network

load is approximately 1500 kBit/s. If the delivery rate is 0% or 100% the network load cannot be determined. How the protocol handles this case will be discussed later.

The node has a second pre-calculated lookup table (see example Table 3.1), in which it can lookup the expected goodput, when the SNR and the network load is known. This table can be created by measuring the goodput of different packet sizes with different network loads and node distances in a topology similar to Figure 3.3.

SNR [dB]	Network Load [kBit/s]	Packet Size [bytes]	Packet Loss Ratio [%]	Goodput [kBit/s]
27	1500	100	0.1	1485.0
27	1500	200	0.2	1470.0
27	1500	300	0.3	1455.0
27	1500	400	0.05	1492.5
….				

Table 3.1 Packet size lookup table

In the given example the 400 bytes achieves the highest goodput and is therefore the *optimum packet* size. The node stores this value in his routing table (SIZEb).

If the delivery ratio of the AODV HELLO-messages is 100% (to be exact: 100% minus packets lost due to frame errors) the node cannot exactly determine the network load. However, it can tell the upper bound of the network load. It then selects the packet size, which is best for the upper bound. If the delivery rate of the AODV HELLO-messages is 0%, the protocol step-wise increases the packet size.

5. Piggy-pack packet size on an AODV HELLO-message

Just like in the Simple Packet Size Selection Protocol the nodes piggy-pack the optimum packet size on AODV HELLO-messages.

6. Store optimum packet size in routing table

When a node receives an AODV HELLO-message it stores the packet size in the corresponding SIZEf entry of its routing table.

3.3 Summary

In this chapter the design of a distributed aggregation mechanism was described. It consists of two (independent) parts – the aggregation queue and the packet size selection protocol. It was argued that the packet size is very important for aggregation since different packet sizes have different packet loss probabilities. The trade-off between too large and too small packets and the causes of packet loss was described. Then it was shown that ETX is not an appropriate metric for deducing the optimum packet size. Because of this, two other methods, one for

calculating the maximum packet size and one for calculating the optimum packet size have been developed. In addition, evidence was given, why these methods need to be distributed protocols, where the receiver calculates the packet size.

4 Implementation in ns-2

This chapter gives an introduction to ns-2 and describes how the Simple Packet Size Selection Protocol and an aggregation queue are implemented in ns-2.

4.1 Introduction to ns-2

Ns-2 is an object oriented *network simulator*, written in C++ and OTcl[5]. It can be used for modelling and simulating network protocols in wired, wireless and satellite networks. The simulator utilizes a technique called *discrete simulation*. Hereby one event after another – for example sending a packet from one node to the next - is processed. The simulation does not run in real time. However all events are captured in a real time like time scale.

The simulator comprises a large number of physical layer models, MAC protocols, routing protocols, network protocols and application traffic generators. Ns-2 is especially useful for the comparison of protocols. It is quite challenging to model a real-life setting in ns-2 and produce the same results as in a test-bed. Nevertheless comparing protocols within ns-2 can be very informative. Sections 4.1.3 and **Fehler! Verweisquelle konnte nicht gefunden werden.** list a few aspects that need to be considered when setting up and analyzing simulations.

The focus of ns-2 is on packet handling, which means that a simulation is the exchange of packets between objects and the processing of the packets by the objects. The next section provides a brief overview on how these objects are implemented and how they are interconnected.

4.1.1 ns -2 Architecture

Ns-2 is a collection of C++ and OTcl classes. The C++ classes are responsible for packet handling and processing. The OTcl classes control the packet flow and provide a configuration interface. This combination for two languages allows a fast processing of the packets by the compiled C++ classes and an interactive simulation setup with the interpreted OTcl classes.

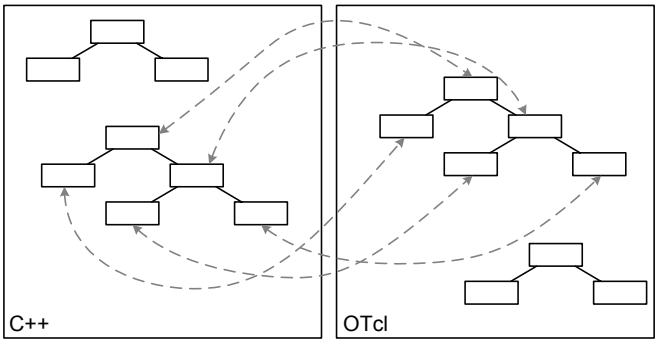

Figure 4.1 Relationship between C++ and OTcl class hierarchies. Source: [65]

The C++ and OTcl class hierarchies are closely related to each other[5]. As Figure 4.1 illustrates, some C++ and OTcl classes are linked together. The OTcl classes are used to manipulate member variables and call control functions of the C++ classes. Some C++ classes that are only used by other C++ classes are not linked to a corresponding OTcl class.

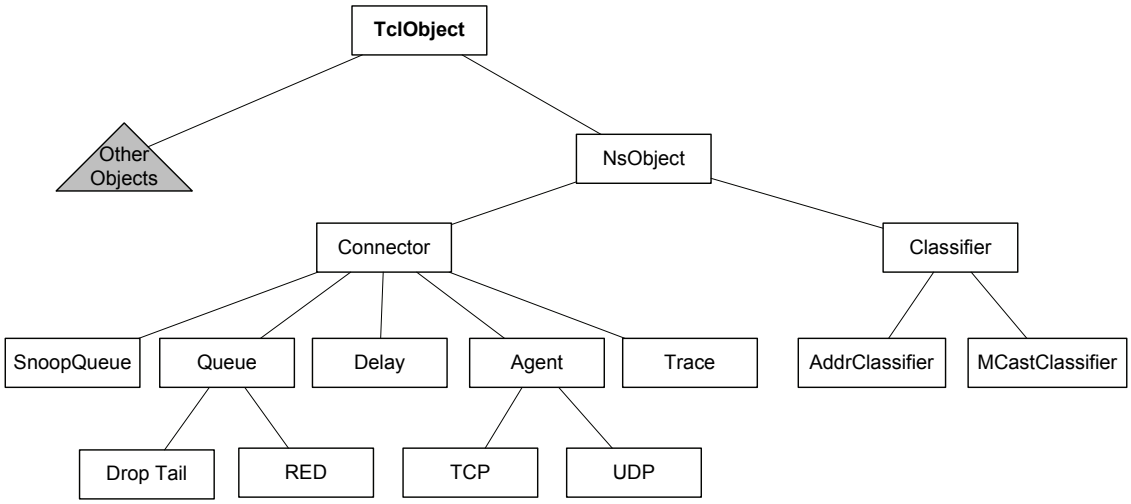

Figure 4.2 Partial ns-2 class hierarchy. Source: [66]

Ns-2 builds up a class hierarchy by the extensive use of inheritance. Figure 4.2 shows a part of this hierarchy. A very important C++ base-class is `NsObject`, which is the superclass to most classes in ns-2. Packets – which are objects themselves - are passed between objects of type `NsObject` or derived types. Objects receive packets with the `recv(...)`-method, process them and forward them to another object by calling the `recv(...)`-method of this object. The `command(...)`-method provides an interface from the OTcl hierarchy. Commands, for example for configuring a C++ object, can be passed from OTcl to a C++ object. The `command(...)`-method evaluates the command parameters and executes the appropriate actions. Another way of exchanging information between OTcl and C++ classes

are bind variables. These types of variables exist both in the OTcl and in the corresponding C++ class. They are mainly used for configuring C++ classes with OTcl.

4.1.2 Basic Components

Ns-2 builds up basic components with the C++ and OTcl classes. A brief description of the basic building blocks is given below. Detailed information can be found in [5].

4.1.2.1 Event Scheduler

As mentioned above ns-2 incorporates a technique called discrete simulation. One event is processed after another. The event scheduler keeps a list of events and execution times for the events.

4.1.2.2 Node Objects

Ns-2 includes different types of nodes, such as a unicast node, a mobile node and a base station node. Figure 4.3 displays the standard node model, the mobile node model and how it is extended for the packet aggregation scheme (also see section 4.2.1). The green parts of the figure were modified and added to implement the aggregation scheme. The blue part was modified in order to create a more realistic simulation environment.

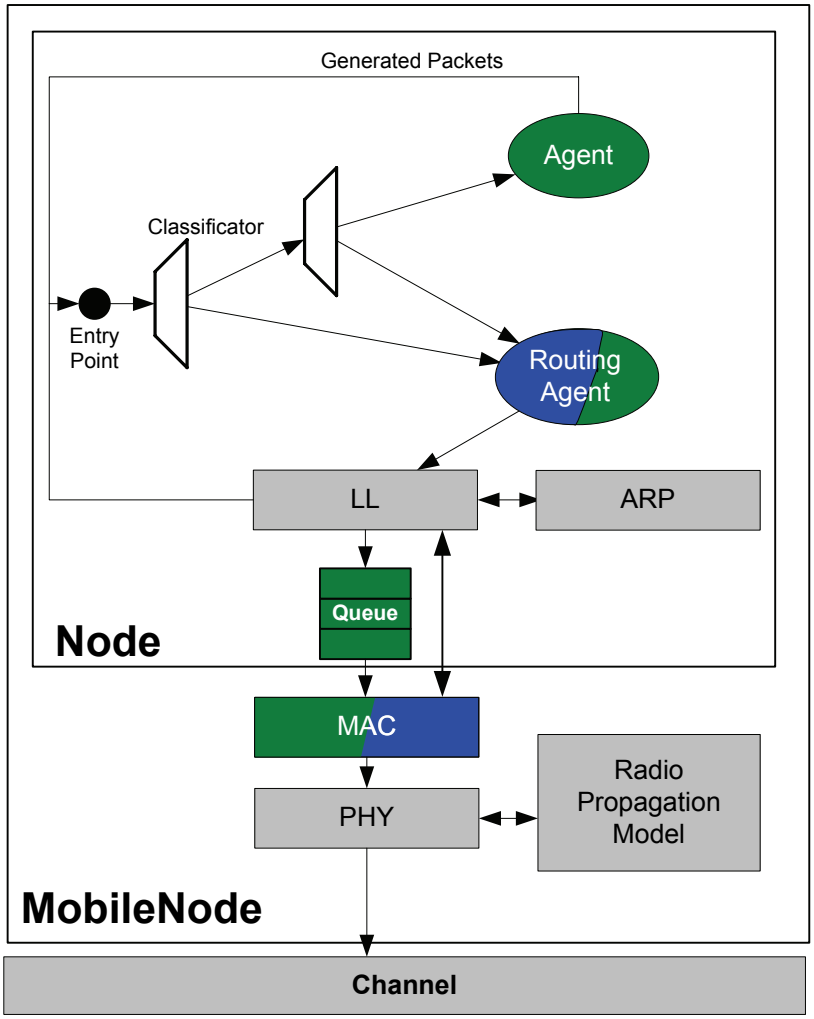

Figure 4.3 Extended and modified node model. Figure based on [5, fig. 16.2]

Each node has an entry point, at which packets arrive. From there the packets are forwarded to classificator objects. A classificator decides the next target of a packet based on the IP destination address and the packet type. Packets, which are destined for the local node, are forwarded to the agent connected with the packet type. The routing agent and the link layer further process all other packets. Subsequently packets are stored in a packet queue. When the MAC layer is idle it request a packet from the queue. The MAC layer sends them down to the physical layer. The Radio Propagation Model calculates the signal strength for a packet. Details are given in section 4.1.2.4. Finally, the channel – which is not part of the node anymore – is responsible for interconnecting nodes with each other.

4.1.2.3 Packets and Queues

Packets are the fundamental units of exchange between objects[5]. They are objects of type `Packet`. A schematic diagram is given in Figure 4.4. Each packet has a pointer to the next

packet either in a queue that it is currently stored in or in the free list, which collects packets that can be deleted from the memory. Beside the `next_` pointer a packet consists of a packet header and packet data. While the data field is usually not used in simulations, the packet header is used for implementing different network protocols. By default, a packet carries headers of all available protocols. Single protocol headers that are not needed in a simulation can be disabled. However, at least the common header is necessary. It stores important information such as the unique packet id and the simulated packet size.

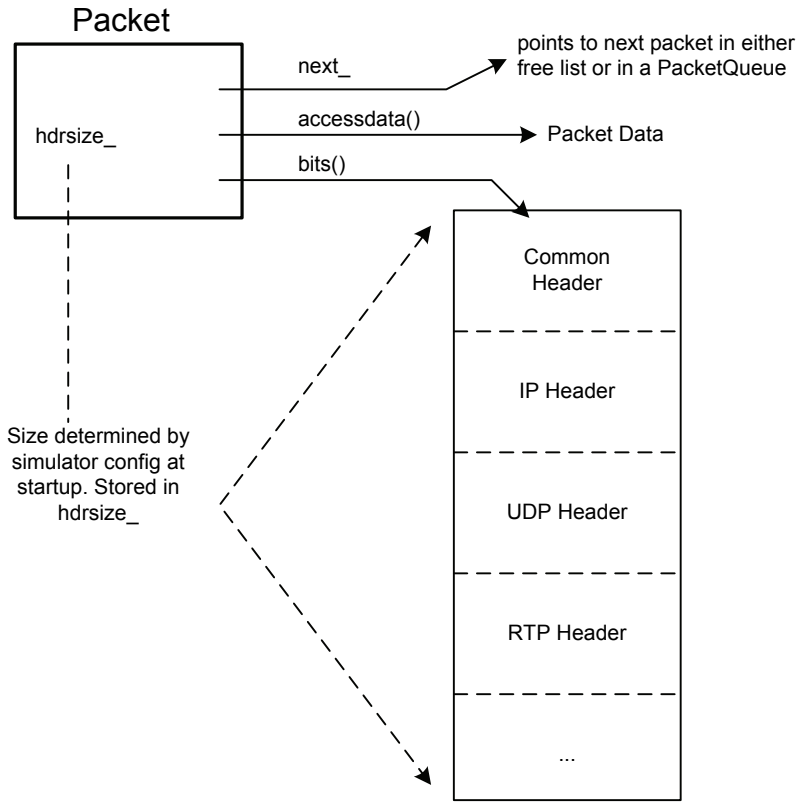

Figure 4.4 ns-2 packet object. Figure based on [5, Fig. 12.1]

Objects of type `PacketQueue` store packets in a linked list. Packets are handed to a queue object via the `enque`-method. When the `deque`-method is called, the queue returns a stored packet. When a queue is currently transmitting a packet to a lower layer, it is blocked. Therefore, it cannot release any further packet. The `PacketQueue` class serves as a base class for the implementation of queues, which are used within the node. In a node, the link layer processes packets and forwards them to the packet queue. They reside in the queue until the MAC layer request a packet by calling `deque()`.

The simplest queue implementation is the `DropTail` queue. It implements a simple FIFO queuing mechanism and drops incoming packets if the internal buffer is full. Ns-2 also offers

more advanced queue types, which for example use class based queuing. Here packets are released in a round-robin manner based on traffic classes and assigned priorities.

4.1.2.4 Physical Layer

A radio propagation model predicts the signal strength of every received packet[5]. For the simulation of wireless channels ns-2 brings three different propagation models. The simplest model is the *free space model*. In this model there is only one clear line of sight path between sender and receiver. The receiving power is dependent on the transmission power, the wavelength, the antenna gain, a system loss coefficient and the distance between sender and receiver. Since this model is not very realistic, a more advanced model – the *two-ray ground reflection model* – is provided too. This model also considers signal reflection on the ground. However, fading – that is the signal strength variation due to multi-path propagation – is not part of the prior models. In the *shadowing model*, which is the third propagation model in ns-2, also fading is part of the signal strength calculation. This model is the most realistic of all three. It can be used to simulate both outdoor and indoor environments.

Configuration Parameter	**Description**
CPThresh_	Capturing threshold
CSThresh_	Carrier sense threshold in dB
RXThresh_	Signal power necessary for receiving a packet, in dB
Pt_	Sending power in mW
freq_	Sending frequency in Hz
L_	System loss factor

Table 4.1 Physical Layer Configuration Parameters

Beside the propagation model, the parameters given in Table 4.1 influence the transmission properties – especially the transmission range and the interference range. They are set by calling

```
Phy/WirelessPhy set parameter value
```

Ns-2 ships a tool[5], which calculates the carrier sense and the receiving threshold for arbitrary distances.

4.1.2.5 MAC Layer

For simulating IEEE 802.11 networks the DCF MAC layer is available in ns-2. `Mac/802_11` implements the RTS/CTS/DATA/ACK scheme described in section 2.1.3.3. When packets are received, the MAC layer is responsible for collision detection and handing the packets over to upper layers. When packets are sent, the MAC layer performs virtual carrier sensing

and sends the packets down to the physical layer. Unsuccessful transfers are re-sent using a binary exponential backoff scheme.

Some parts of the MAC layer behaviour can be configured with OTcl variables, others with pre-processor statements in "mac/mac-802_11.h". The OTcl variables `basicRate_` and `dataRate_` set the speed for transmitting control frames and data frames. In "mac-802_11.h" pre-processor macros control the MTU size, the use of RTS/CTS, the retransmission behaviour and the MAC layer timing. By changing these settings different standards such as IEEE 802.11b or IEEE 802.11g can be simulated. [61] lists the required settings for simulating an IEEE 802.11b channel. For example the `dataRate_` is 11 Mbit/s, the `basicRate_` is 1 Mbit/s, the `freq_` is 2.472 GHz and the sending power `Pt_` is 31 mW.

4.1.2.6 Routing

In ns-2 the routing functionality is split up into three parts[5, 66]:

- Routing Agent: exchanges routing information with neighbouring nodes
- Route Logic: calculates the route table based on the routing information exchanged by the agents
- Classifiers: perform the actual packet forwarding based on the route table and the packet destination/next hop

Each of those parts can be implemented independently. For implementing a new protocol it might only be necessary to create a new routing agent and use existing modules for route calculation and packet forwarding. Ns-2 ships a number of wireless routing protocols, such as DSR and AODV. The implementations do not use advanced routing metrics such as the ETX.

4.1.2.7 Agents and Traffic Generators

Agents are the communication endpoints in ns-2. They generate and consume packets. In the simulation setup agents are attached to the nodes. The simulator provides different ways of generating traffic. One way is the use of a *Transport Agent*. Transport Agents are mainly used for TCP. They have a programming interface similar to sockets. For example, the function call `send(1000)` causes an agent to send 1000 bytes via TCP. On top of Transport Agents the *Application Agents* can simulate whole applications like Telnet or FTP. The setup of an ftp session is illustrated in Listing 4.1. In lines 1-5 the Transport Agents are generated and connected. In lines 8 and 9 the Transport Agent are connected to an FTP Application Agent.

[5] It can be found in "indep/threshold" within the ns-2 source directory.

```
001: set src [new Agent/TCP/FullTcp]
002: set sink [new Agent/TCP/FullTcp]
003: $ns_ attach-agent $src-node $src
004: $ns_ attach-agent $dst-node $sink
005: $ns_ connect $src $sink
006: $sink listen
007: $src set window_ 100
008: set ftp1 [new Application/FTP]
009: $ftp1 attach-agent $src
010: $ns_ at 0.0 "$ftp1 start"
```

Listing 4.1 Setup of an FTP connection in ns-2

A very flexible method for traffic generation, also VoIP traffic, is the use of a *Traffic Generator* object. Ns-2 brings along four types of traffic generators:

- Application/Traffic/Exponential: generates packets in ON periods and no packets in OFF periods. The ON/OFF periods are exponentially distributed.
- Application/Traffic/Pareto: Similar to the exponential traffic generator, but the ON/OFF periods follow a Pareto distribution.
- Application/Traffic/CBR: constantly generates packets of a configured size with a specified data rate
- Application/Traffic/Trace: generates packets based on the information provided in a trace file

Traffic generators periodically call the attached agent, for example `Agent/UDP`. This agent generates a packet and sends to the node entry. The size of the packet can be configured with the `packetSize_` variable, the destination address is dependent on the attached receiver sink.

```
001: set src [new Agent/UDP]
002: set sink [new Agent/Null]
003: $ns_ attach-agent $src-node $src
004: $ns_ attach-agent $dst-node $sink
005: $ns_ connect $src $sink
006: set e [new Application/Traffic/Exponential]
007: $e attach-agent $src
008: $e set packetSize_ 32
009: $e set burst_time_ 350ms
010: $e set idle_time_ 650ms
011: $e set rate_ 12.8k
012: $ns_ at 0.0 "$e
```

Listing 4.2 Simulation of a VoIP connection in ns-2

Listing 4.2 exemplifies the setup of a VoIP connection. In lines 1-5 the transport agents are generated and connected. In lines 6-12 the exponential traffic generator in created, configured and started.

4.1.3 Simulation Process

Performing a simulation with ns-2 usually follows a certain process, which is illustrated in Figure 4.5. The three main steps are the simulation setup, running the simulation and analyzing the results.

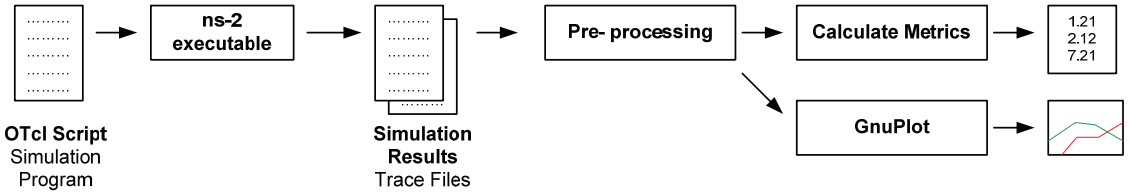

Figure 4.5 ns-2 simulation process

4.1.3.1 Simulation Setup

OTcl scripts set up simulation scenarios. Usually they consist of the following parts[66]:

- Creation of global simulation objects like the simulator instance, the god object (a global management object) and the trace file objects
- Configuration of nodes, i.e., setting of parameters that will be used when nodes are created
- Creation of nodes
- Configuration and creation of agents and applications
- Scheduling of events, e.g. starting or stopping of a traffic generator or node movement
- A finish procedure, which is scheduled to be called at the end of simulation. In this procedure the trace files are flushed
- Starting of the simulation

How a simulation scenario is set up in detail is very much dependent on the purpose of the simulation. Chapters **Fehler! Verweisquelle konnte nicht gefunden werden.** and **Fehler! Verweisquelle konnte nicht gefunden werden.** describe how the simulation for the performance evaluation of the packet aggregation is set up.

4.1.3.2 Running the Simulation

The ns executable loads the simulation script and interprets the simulation script. The runtime of the simulation is dependent on the number of nodes, the traffic, the propagation model, the simulation time and the machine ns-2 is running on.

4.1.3.3 Analyzing the Results

When running a simulation ns-2 logs events together with the associated packet ids, nodes ids and a timestamp in a *trace file*. Ns-2 uses different trace file formats. The old trace format for wireless simulations, which is used by default, looks like that:

```
D 1.338127300 _3_ MAC  ERR 32 cbr 147 [a2 2 1 800] ------- [1:0 2:0 32 2] [27] 0 0
s 1.338137133 _2_ MAC  --- 0 ACK 29 [0 1 0 0]
r 1.338152133 _2_ AGT  --- 32 cbr 80 [a2 2 1 800] ------- [1:0 2:0 32 2] [27] 1 0
```

The bold-faced part is produced for every event; the normal-faced part is only added for some payload types. The optional part provides information about IP addresses and ports. Table 4.2 describes the structure of the old wireless trace format. A more detailed description of all trace file formats can be found in [5] and [67].

Column	Name	Description
1	Event Type	"r" means received, "s" sent, "D" dropped and "f" forward
2	Time Stamp	Event time
3	Node ID	Source of event
4	Trace Source	either MAC layer (MAC), router (RTR) or agent (AGT)
5	Reason	indicates that reason for the event, e.g. "ERR" means drop due to frame errors
6	Packet UID	is unique for every packet
7	Packet type	is dependent on traffic generator
8	Packet size in bytes	
9	Transmission time	Expected transmission time on MAC layer
10	MAC address of source	
11	MAC address of destination	
12	Type	

Table 4.2 ns-2 old wireless trace format

When many nodes and a lot of traffic are involved, several hundred megabytes of trace files can be produced per simulated minute. Therefore the trace files are normally pre-processed with Perl or awk-scripts. With these scripts certain metrics, like packet loss ratio are extracted. For visualizing the results tools like Network Animator, GnuPlot or Microsoft Excel are suitable.

When analyzing and interpreting the results the following aspects should be taken into account:

- The simulated runtime has an effect on the exactness of the results. Longer simulation runtimes produce more samples and have therefore a better statistical significance.

- When using random numbers to generate random behaviour the simulation is not deterministic. Each simulation run will produce different results. Fixed seeds for the random number generator can be used to generate pseudo random, but repeatable behaviour. Another option is to run a series of simulations with random seeds and calculate the average of the results.
- The simulation results are only as good as the simulation model. If unrealistic models are used the results might not correspond with the real world.

4.2 Implementation in ns-2

The implementation of the packet aggregation scheme in ns-2 consists of

- an extended and modified node model,
- the implementation of the Simple Packet Size Selection Protocol,
- a new packet type,
- an aggregation queue,
- the deaggregator agent,
- and some scripts to start the simulation and analyze the results.

Table A.1 in the appendix lists the corresponding source code files of the single components.

4.2.1 Extended Node Model

One option for implementing packet aggregation in ns-2 is to modify and extend the mobile node model. The green parts of Figure 4.3 show the parts of the standard mobile node object, which were modified in order to implement the aggregation. The blue parts were modified to achieve a more realistic and stable simulation environment:

4.2.1.1 Frame Error Calculation

By default, ns-2 does not take into account the frame length when determining, if a frame is received correctly or not. Instead the decision is based on three thresholds [65]. On the physical layer it checks whether the signal strength of the incoming packet is above the *carrier sense threshold* `CSThresh_`. If the signal strength is below the carrier sense threshold the packet is discarded and not processed further. If the physical layer senses the packet correctly, the MAC layer verifies that the signal strength is above the *receiving threshold* `RxThresh_`. Otherwise the packet is dropped. If multiple packets are received at the same time, the MAC layer tries to process the packet with the strongest signal. The other packets

are regarded as noise. If the signal noise ratio is above the *capture threshold* `CPThresh_`, the packet can be decoded correctly.

All these calculations do not include the frame length. However, analytical models and experiments [58, 59] show that the frame length influences the *frame error rate* (FER) significantly. Beside this, the determination of the optimum frame size is an integral part of packet aggregation. Therefore a patch [61] was applied to the ns-2 MAC and PHY layer. The patch calculates the bit error rate based on the signal-to-noise-ratio and the transmission rate using the formula given in section 3.2.2.

4.2.1.2 AODV-UU

The AODV implementation of ns-2 supports only pure wireless simulations. In order also to conduct wireless-cum-wired simulations an alternative AODV implementation by Uppsala University[66] was used. AODV-UU supports mixed scenarios, with both wired and wireless nodes. Furthermore, it provides extensive tracing support.

For conducting the ETX experiments described in section 3.2.3.1 an extended version of AODV-UU provided by Karlstad University was used[68]. This AODV-UU extension calculates the ETX for next-hop neighbours. Moreover it allows to base the route selection on least cumulative ETX instead of least hop count.

4.2.1.3 β-Error Routing Patch

In the default AODV-UU implementation a route rediscovery is triggered after a packet is lost and the route is marked spoiled. This can also happen if the contention level is high, but not alternative route is available or the current route has the best link quality. As [69] points out, a route rediscovery leads to additional maintenance traffic, which causes network overload and more errors due to congestion. [69] proposes a scheme, in which a route rediscovery is only triggered after β consecutively lost frames. Using this β-factor fewer route errors are produced and a higher throughput can be gained. This method works especially well in static multi-hop networks. Also mobile nodes benefit from it; however the time needed for successfully re-establishing a route which was destroyed by mobility increases.

For the simulation environment this scheme was adapted to AODV-UU. The AODV-UU agent includes a counter of lost packets. After each successfully transmitted packet the counter is reset. If the counter exceeds the β-factor a route rediscovery is triggered. [57] gives an example in which the number of dropped packets decreases by almost 80% when the patch is applied and aggregation is used.

4.2.2 Simple Packet Size Selection Protocol

The implementation of the Simple Packet Size Selection Protocol consists of one part, which measures the signal-to-noise ratio, and one part, which propagates this information to the neighbouring nodes.

4.2.2.1 Maintain SNR information and calculate packet size

In the MAC layer the `recv_timer()` method is called when a packet is received. In this method the SNR for every unicast data packet is calculated (see Listing 4.3). The signal strength, the noise and the interference level can be obtained from the physical layer and the propagation model.

```
001: snr=10*log10(signal/(noise+interference));
```

Listing 4.3 Calculation of SNR

In order to calculate the smoothened SNR for the backward link, the previous hop of the packet needs to be found (see Listing 4.4, lines 1-3). It is obtained by looking at the next hop for the source address of the packet, which can be found in the routing table (line 8). For each backward link the SNR is stored in the SIZEb field of the routing table. In addition, the maximum packet size is calculated using the equation from section 3.2.3.2 and stored in the SIZEb entry (lines 10-12).

```
001: if ((rtentry_src=aodvagent->rt_table_find(src)) != NULL) {
002:     prev_hop=rtentry_src->next_hop;
003:     if ((rtentry_neighbour=aodvagent->rt_table_find(prev_hop)) !=
NULL){
004:        float snrold, snrnew, sizenew;
005:        snrold=rtentry_neighbour->snr_backward;
006:        snrnew=snrold+0.2*(snr-snrold);
007:        if (snrnew<0) snrnew=0.0;
008:        rtentry_src->snr_backward=snrnew;
009:        rtentry_neighbour->snr_backward=snrnew;
010:        sizenew=0.0035*exp(snrnew*1.2255);
011:        rtentry_src->size_backward=sizenew;
012:        rtentry_neighbour->size_backward=sizenew;
013:     }
014: }
```

Listing 4.4 Store smoothed SNR in the routing table

4.2.2.2 Propagate maximum packet size

Every node periodically broadcasts AODV HELLO messages by calling the `hello_send(…)` method. AODV-UU has an extension mechanism, which allows adding user defined fields to those messages. Using this mechanism the next hop and the

corresponding SIZEb routing table entry is attached to the AODV HELLO message (see Listing 4.5).

```
001: for (j = 0; j < RT_TABLESIZE; j++) {
002:   list_t *pos;
003:   list_foreach(pos, &rt_tbl.tbl[j]) {
004:      rt_table_t *rt = (rt_table_t *)pos;
005:      memcpy(AODV_EXT_NEXT(ext), &rt->dest_addr, sizeof(struct
in_addr));
006:      ext->length+=sizeof(struct in_addr);
007:      memcpy(AODV_EXT_NEXT(ext), &rt->size_backward, sizeof(float));
008:      ext->length+=sizeof(float);
008:   }
008: }
```

Listing 4.5 Adding the address/SNR pair to the AODV HELLO message

The piggy-packing of the SNR information can be enabled and disabled by calling

`Agent/AODVUU set modesnr_ true/false`

from the simulation script.

When a node receives an AODV HELLO it calls the `hello_process(…)` method. There it reads the maximum packet size field for its address and stores it in the SIZEf routing table entry for the source of message.

4.2.3 Packet type IP_META

A new packet header type IP_META is used to transport the aggregated packets from node to node. This packet type implements the encapsulation technique illustrated in Figure 3.1 by deriving a class from the `PacketHeader` class and creating a header structure `hdr_ipmeta` (see Listing 4.6).

```
001: struct hdr_ipmeta {
002:    void encap(Packet** p, int numpkts) {
003:       for (int i = 0; i < numpkts; i++)
004:         ep[i] = p[i];
005:       size_  = numpkts;
006:    }
007:    Packet **decap()  {
008:       return ep;
009:    }
010:    int size() { return size_; }
011: private:
012:    Packet *ep[MAX_PACKET_NUM];
013:    int size_;
014: };
```

Listing 4.6 IP_META header definition

It includes an array of pointers (`ep*`) to the aggregated packets. The class `Packer` provides helper functions to create and unpack IP_META meta packets. The function `Packet* pack(Packet** p, int numpackets, int destination)` creates a new packet of type IP_META.

4.2.4 Queue/Aggregator/Adaptive

The aggregation queue is one core part of the packet aggregation scheme. It is derived from the `PacketQueue` class. The corresponding OTcl class is `Queue/Aggregator/Adaptive`.

enque(…) method

The link layer passes packets to the queue by calling the `enque(…)` method (see Listing 4.7). Incoming packets are marked with a timestamp and stored in an internal packet queue (lines 4,5). If the internal packet queue is full, all newly incoming packets are dropped (line 8).

```
001: void AggregatorAdaptive::enque(Packet* p) {
002:    struct hdr_cmn *ch = HDR_CMN(p);
003:    if (ch->ptype() == PT_EXP || ch->ptype() == PT_ENCAPSULATED)
004:      ch->ts_ = Scheduler::instance().clock();
005:    q_->enque(p);
006:    if (q_->length() > qlim_) {
007:       q_->remove(p);
008:       drop(p);
009:    }
010: }
```

Listing 4.7 enque(…)-method

deque() method

The deque() method implements the actual aggregation algorithm. It is either called when

- the MAC layer is idle or
- a new packet is en-queued or
- the delay timer of a queued packet expires

The aggregator queue only aggregates packets of type EXP, which are used to simulate VoIP traffic. All other packets are simple passed through in a FIFO manner. Other packet types which should be aggregated can be added in line 4 of Listing 4.8.

Figure 4.6 shows a flow-chart of the aggregation algorithm. At first, the size of all packets that can be aggregated is calculated. Those are all packets, which have the same next hop as the oldest packet P_0.

```
002: Packet* AggregatorAdaptive::deque() {
003:     …
004:     if ((hdr_cmn_packet->ptype() == PT_EXP || hdr_cmn_packet-
>ptype() == PT_ENCAPSULATED)) {
005:         …
006:     }
007: }
```

Listing 4.8 Excerpt of the deque()-method

If the size is equal or greater than the $SIZE_{min}$ threshold, a new IP_META packet is generated (left branch) and sent. All packets with the same next hop as P_0 are aggregated in it. The size of the IP_META packet must not exceed the $SIZE_{max}$ threshold and the MTU.

If the calculated size is less than $SIZE_{min}$ the right branch of the flow chart is executed. Here all packets older than T_{delay} are aggregated. There is no minimum size this time, but the maximum size of IP_META is again determined by $SIZE_{max}$. If more than one packet is older than T_{delay} an IP_META packet is sent. If only one packet is older, the packet is sent as it is. There is no reason to add an IP_META header. If no packet is older than T_{delay} nothing is sent and a timer is set, so that the `deque()` method is called when P_0 is older than T_{delay}.

If the followup parameter is set to true, the queue is always flushed when the right branch is executed. This can be useful to reduce the delay for some packets. However, it could increase the jitter, since the aggregation is not as steady then.

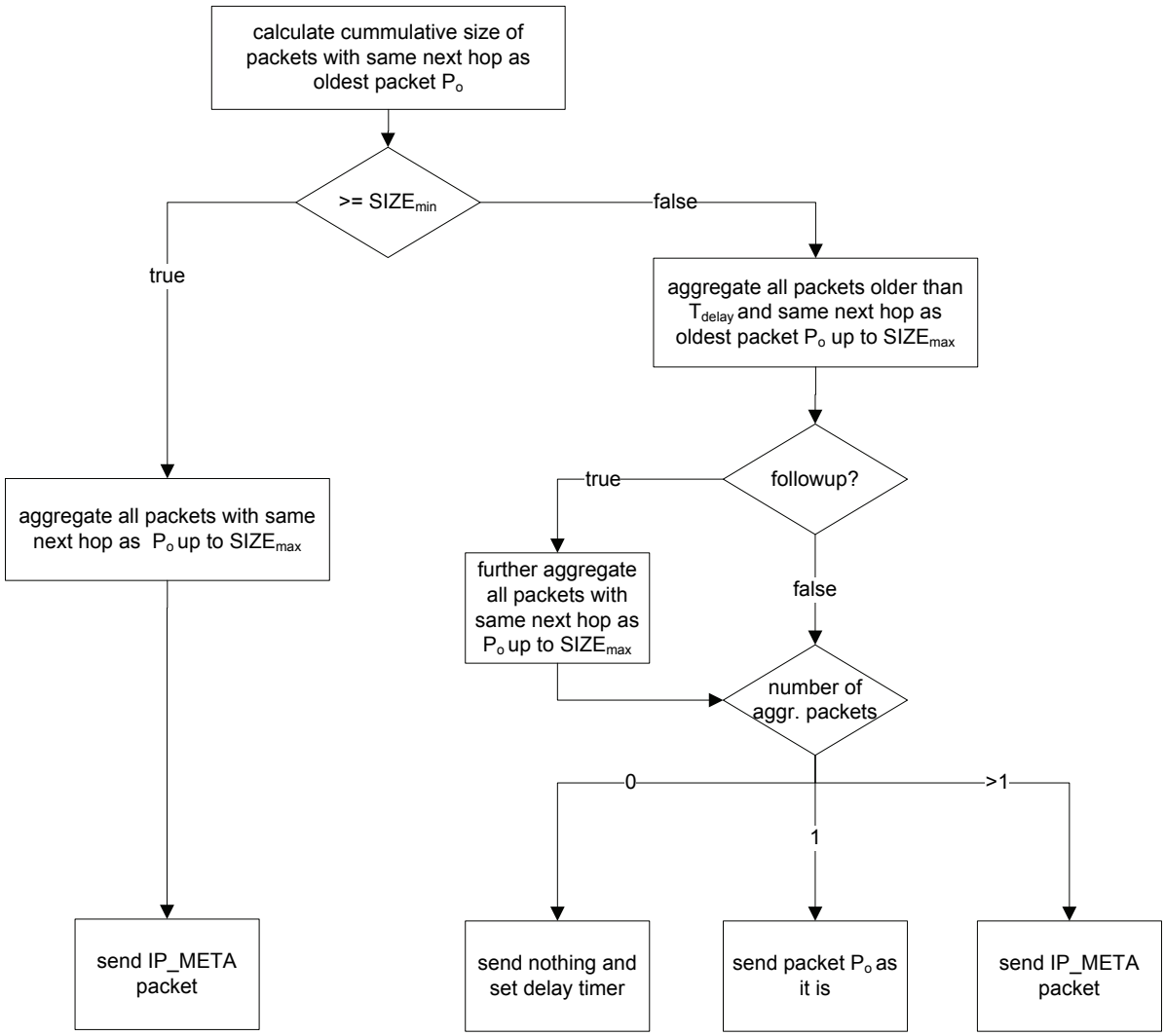

Figure 4.6 Flow chart of the aggregation algorithm

Selection of $SIZE_{min}$ and $SIZE_{max}$

$SIZE_{min}$ and $SIZE_{max}$ are the two foremost configuration parameters of the aggregation algorithm. $SIZE_{min}$ defines the minimum size an IP_META packet has to have. If it is too small, the additional IP-header for the IP_META packet generates too much overhead. If it is too large, many packets might pass through un-aggregated. An analysis of the effects of $SIZE_{min}$ will be given in section 5.2.5.

The $SIZE_{max}$ parameter defines the maximum possible packet size. It is calculated by the Simple Packet Size Selection Protocol and an additional parameter sizefactor_. Sizefactor_ can be used to increase the packet size and therefore influence the ratio of packet loss due to frame error and packet loss due to contention. The actual $SIZE_{max}$ is calculated as $SIZE_{\max} = SIZE_{\max snr} * sizefactor\_$, where $SIZE_{maxsnr}$ denotes the maximum packet size

calculated upon the SNR by the Simple Packet Size Selection Protocol. $SIZE_{maxsnr}$ denotes a packet size that creates less than 0.2% packet loss per hop due to FER. However, aggregation produces many packets, which are smaller than $SIZE_{maxsnr}$ and have a lower loss ratio due to FER. Therefore the actual maximum packet size can be increased by the sizefactor_ without producing too high packet loss.

OTcl Configuration Variables

The aggregator queue can be configured from the simulation script by calling

```
Queue/Aggregator/Adaptive set parameter value
```

before creating a node object. The available configuration variables are listed in Table 4.3

Configuration Parameter	Description
adaptive_mode_	If activated $SIZE_{max}$ is determined adaptively. Otherwise $SIZE_{max}$ equals the MTU-size. This case is called "*static aggregation*".
agg_followup_	"true" activates the followup branch
aggthreshold_	Configures $SIZE_{min}$
debug_output_	Enables debug output to stdout
max_delay_	Configures the T_{delay} parameter
mtusize_	MTU size of the underlying MAC protocol.
sizefactor_	Factor controlling loss due to frame errors and contention

Table 4.3 Configuration parameters of Queue/Aggregator/Adaptive

4.2.5 Agent/Deaggregator

The agent `Agent/Deaggregator` receives packets of type IP_META, unpacks them and forwards them to the node entry point (see Figure 4.3). When nodes process packets, classificator objects decide where to route a packet. Based on the destination address or packet type packets can be forwarded to different agents. In order to activate the routing of IP_META packets to the Agent/Deaggregator the agent needs to be attached to the node in the simulation setup by the following command:

```
$node attach [new Agent/Deaggregator] 58
```

Here, 58 represents the numerical identifier or the IP_META packet type. This number can differ, if additional packet header types are available in the ns-2 installation.

4.2.6 Simulation and Analyzer Scripts

To efficiently perform and analyze a large number of simulations different scripts were written. Figure 4.7 show the simulation environment that the scripts build up. The simulation control script `runtest.sh` is a bash script. It starts single simulation runs and calls results analyzer script. When calling the simulation script `simulation.tcl` and the ns executable

the control script passes over parameters, which for example determine the distance between nodes. The control script can start several successive simulation runs with different parameters (e.g. to simulate different node distances). When a simulation run is done, the control script calls the analyzer script.

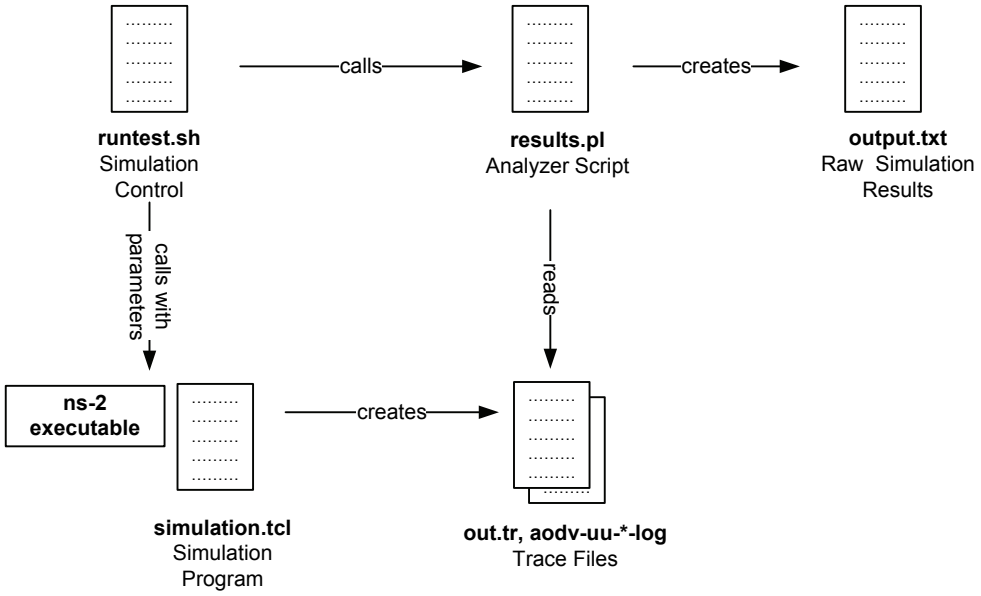

Figure 4.7 Simulation Architechture

The analyzer script `results.pl` is a Perl script. It reads the trace files which were generated by ns-2. The script reads the trace files line by line and counts events, calculates packet delays etc. The script works with the wired and the wireless trace format. In order to allow the separate analysis of single VoIP flows, the wireless trace file format was extended with an extra column, that includes the flow-id.

5 Simulations and Evaluation of the Algorithm

This chapter describes how the simulation environment was set up to evaluate the aggregation algorithm. It defines performance metrics and analyses the performance of the adaptive algorithm. The results are compared to no aggregation and static aggregation. *Static aggregation* means that the $SIZE_{max}$ parameter equals the MTU of the link. Beside this, the static aggregation algorithm is identical to the adaptive aggregation algorithm. Two different simulation topologies – the *Arrow Topology* and the *Grid Topology* – were used to study the packet aggregation in detail.

5.1 General Simulation Setup

The most important settings which were used in the simulations are listed in Table 5.1. If a setting was changed in a single simulation this is explicitly noted in the respective simulation description.

The VoIP traffic was simulated by sending 40 byte UDP packets at a rate of 50 packets per second with an on/off interval of 350/650 ms. This represents an average VoIP flow with voice activity detection[70] and G.729a. In the simulations several flows were active at once. To avoid traffic peaks the flows were started with a 70 ms time difference. The traffic generation stops shortly before the simulation end. No background traffic for simulating a signalling protocol was in place.

Name	**Value**	**Note**
General Settings:		
Simulation Time	120 sec	
MAC/Physical Layer:		
Receiving Threshold	$4.496*10^{-12}$ dB	67 m of transmission range
Carrier Sense Threshold	$8.29214*10^{-13}$ dB	134 m interference range
Sending Power	30 mW	
Sending Frequency	2.4 GHz	IEEE 802.11b
Data Rate	11 Mbit/s	
Basic Rate	1 Mbit/s	
RTS/CTS	Off	
Propagation Model	Shadowing	
Shadowing Derivation	1.1 dB	
Shadowing Path Loss Exponent	2.5	to simulate an outdoor environment [5]
Routing:		
Routing Protocol	AODV-UU	

ALLOWED_HELLO_LOSS	5	Invalidates route only after 5 non-received HELLO-Messages. Increases route stability.
β-factor	2	Trigger a route error after 2 consecutive packet drops[69].
Traffic Generation:		
Traffic Generator	Exponential	
Packet Size	32 bytes	20 bytes audio payload + 12 bytes RTP-header
Busy/Idle Interval	350/650 ms	Average VoIP call with Voice Activity Detection
Traffic Rate per flow	12.8 kbit/s	G.729a send rate
Aggregation Queue:		
T_{delay}	5 ms	
$SIZE_{min}$	101 bytes	
Followup	false	

Table 5.1 Simulation Settings

The topologies were designed to have stable routes. For example a node distance of 50 m gives a good channel quality. In a three-node string (N0←→N1←→N2) with 50 m distance each, the left-most node N0 would usually communicate with the right-most N2 node via the middle node N1. However, it can happen that an AODV-HELLO message from N0 is received by N2 correctly. Since this is the shortest path, the N2 wants to communicate with N0 directly. However, over a distance of 100 m the packet loss is very high. To solve this problem the propagation parameters and the node distances were chosen in a way, such that N2 cannot receive an AODV-HELLO message from N0.

5.2 Arrow-Topology

5.2.1 Topology description

The first simulation topology is sketched in Figure 5.1. Due to its arrow-shape it will be denoted by "Arrow-Topology" subsequently. The topology comprises wireless and wired nodes. *Node 0* represents a server, e.g. a PSTN-gateway. It is connected with a Fast-Ethernet (100 Mbit/s) to the router *Node 1*, which itself has an wired Ethernet connection to *Node 2*. *Node 2* is a Mesh Access Point, which has both, a wired and a wireless network interface.

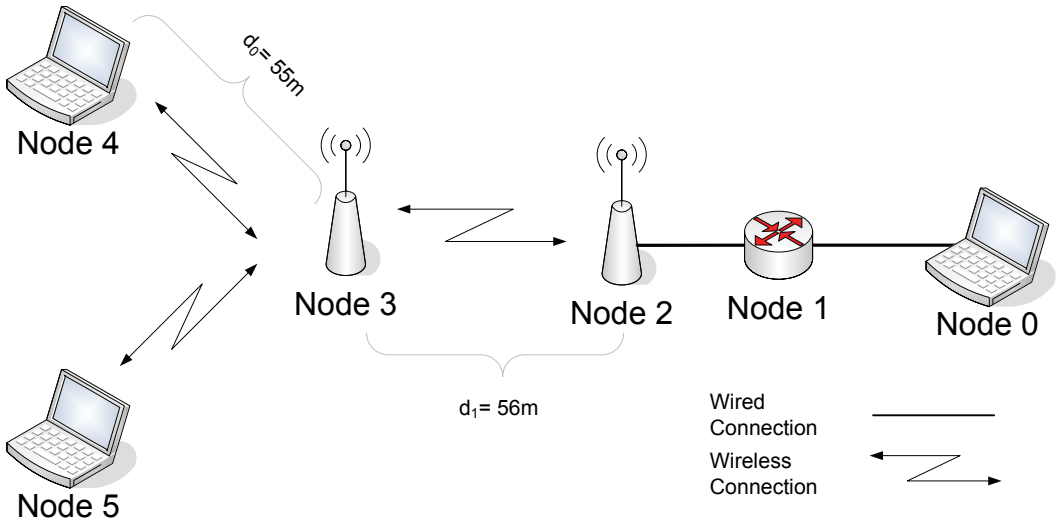

Figure 5.1 Arrow Topology

Node 3 is a Mesh Point, which only forwards traffic. *Node 4* and *5* are Mesh Clients. The RTP-streams are set up between *Nodes 5* and *0*, *Nodes 4* and *0* and the respective reverse direction. All nodes with wireless network interfaces are capable of aggregating and de-aggregating traffic. All nodes are stationary. The distance between them are d_0 and d_1. The values of d_0 and d_1 are 55 and 56 m and are varied in some simulation runs.

The topology presented in Figure 5.1 incorporates two wireless hops. It can be extended to *n* wireless hops by adding mesh points between Node 2 and Node 3. The distance between Node 2, Node 3 and the additional nodes is d_1 for each hop.

5.2.2 Simulation 1 – General Evaluation

The first simulation should help to evaluate the capacity and performance in terms of supported flows[6] for no aggregation, static aggregation and adaptive aggregation. Therefore series of simulations were conducted, whereas in each simulation the number of injected flows (that means the number of flows generated at the sender) was increased.

[6] One call of a flow in forward and a flow in reverse direction. Flows are used instead of calls here to have a more fine-granular analysis.

5.2.2.1 General comparison

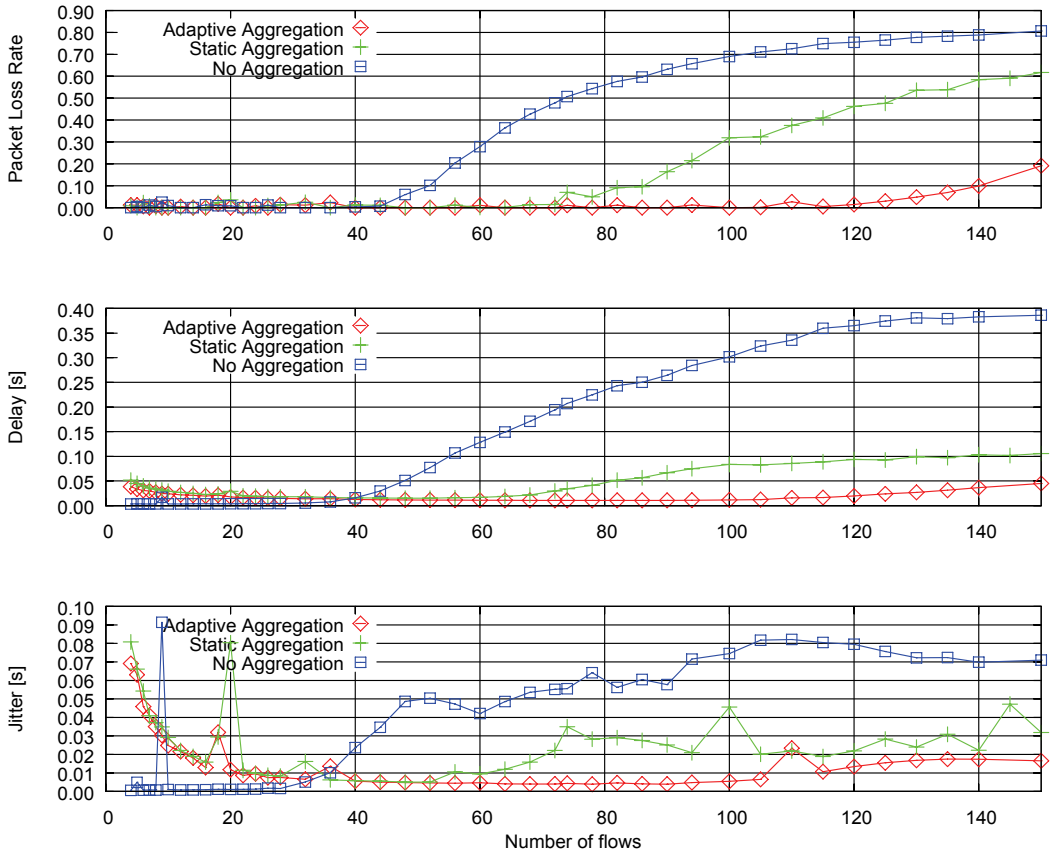

Figure 5.2 Comparison of adaptive aggregation, static aggregation and no aggregation

Figure 5.2 shows the packet loss ratio, average delay and average jitter for adaptive aggregation, static aggregation and no aggregation. The values were obtained by increasing the number of injected flows with each simulation run and then calculating the average packet loss ratio, delay and jitter for all flows in the current simulation. The packet loss rate does not include packets lost due to too high jitter or delay.

The diagram shows that up to a certain threshold the number of injected flows can be increased, while the QoS parameters stay within boundaries. Above this threshold packet loss, delay and jitter rise immediately. With no aggregation this threshold is about 40 flows, for static aggregation it is about 60 flows, for the adaptive method it is about 120 flows. However, average values do not provide enough information to assess, whether a single flow fulfils the quality requirements. Therefore a more detailed analysis was made:

5.2.2.2 Number of supported flows

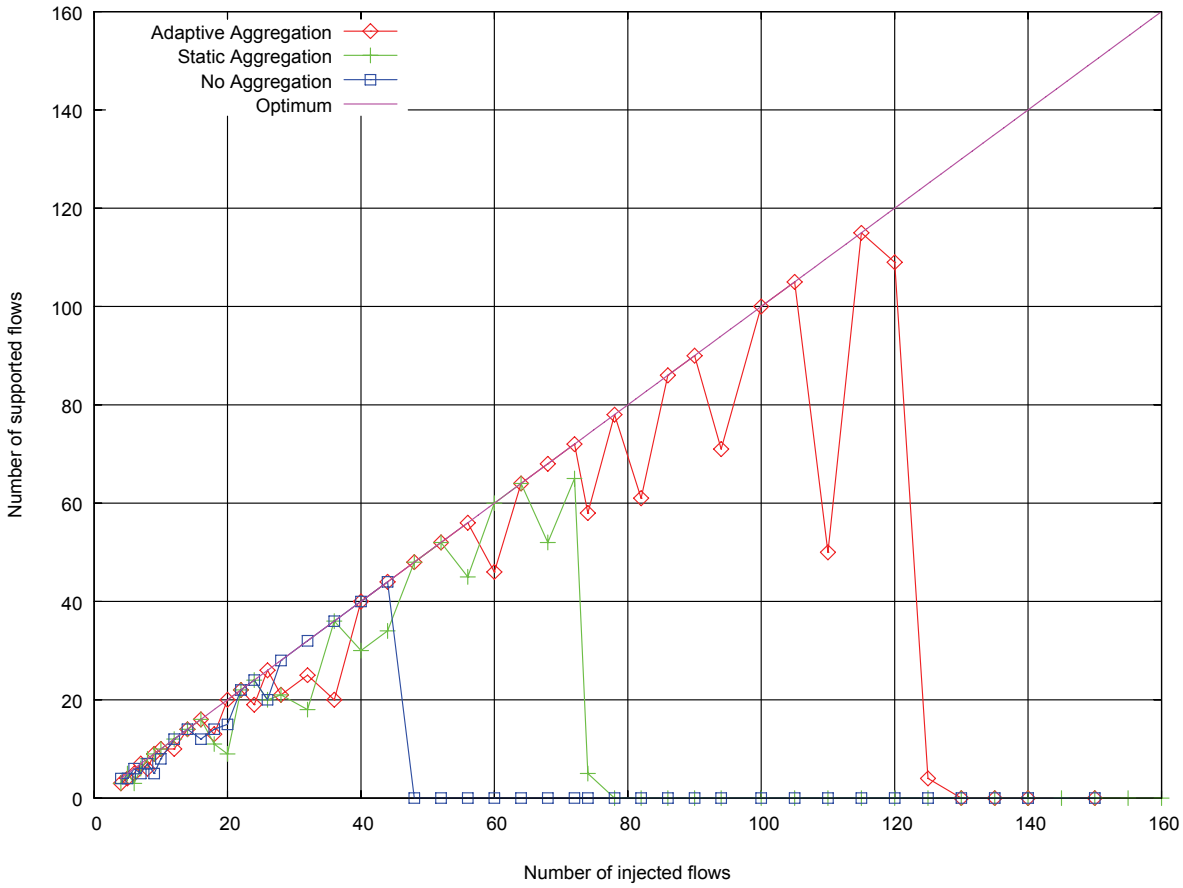

Figure 5.3 Supported flows

Figure 5.3 displays how many flows actually fulfil the QoS requirements for VoIP. For each flow those parameters were analyzed separately. A flow is regarded as "supported flow", if the average packet loss ratio is below 2% and the sum of jitter and delay is below 150 ms. The clients use a "perfect" adaptive jitter buffer, which varies its size dependent on current jitter. By Figure 2.12 2% packet loss ratio and 150 ms end-to-end delay should result in an satisfactory call quality when G.729a is used. Other codes such as G.711 might allow higher packet loss ratios.

In Figure 5.3 a threshold for maximum number of supported flows can be seen very clear. With no aggregation the number of injected flows can be increased up to 42, while almost all flows provide good quality. But if only two more flows are added, no flow provides good quality. By using static aggregation this threshold can be improved to 68 flows, with adaptive aggregation it can be enhanced to 115 flows (why adaptive aggregation supports more flows than static aggregation will be discussed in section 5.2.3). Further on the threshold beyond

which the rate of supported flows drops from almost 100% to 0% will be called the *VoIP capacity* of the topology.

Nevertheless, even below the capacity threshold it happens that some flows have bad quality[7] and are therefore counted as "not supported". Ideally, all flows up to the threshold should be supported (which is represented by the pink line). For example, with adaptive aggregation, of 94 injected flows only 71 are supported.

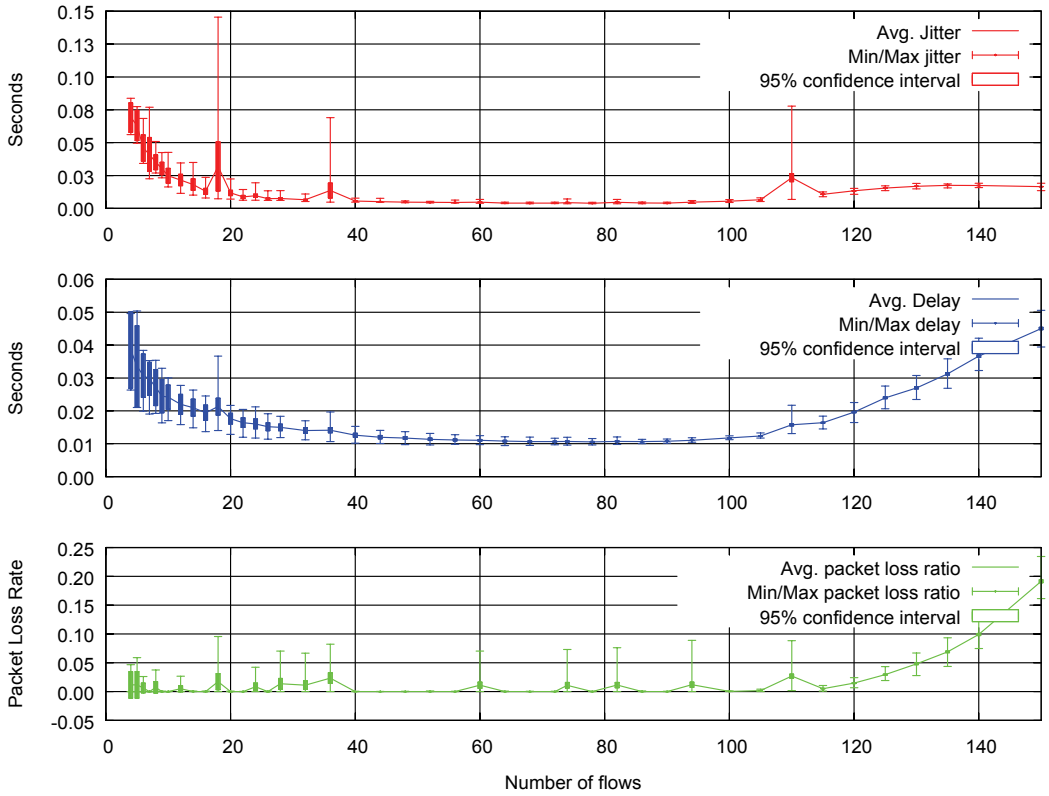

Figure 5.4 Packet Loss, Delay and Jitter for Adaptive Aggregation

Figure 5.4 shows what causes this phenomenon. In this diagram not only the average values over all flows, but also the minimum, the maximum and the 95% confidence interval of the average values of the single flows are plotted. The small confidence intervals indicate that the quality of most flows is close to the average quality. For few flows the confidence interval is high, because of too little samples (e.g. only 4 samples with 4 flows).

The maximum values vary quite strongly from the average. That means that the network could still support all flows (because the average is good enough), but due to flow unfairness

[7] Quality is used as an umbrella term for packet loss ratio, delay and jitter here.

some flows have worse quality than others. However, the reason for this flow unfairness is not obvious.

Figure 5.4 and Figure 5.2 show that aggregation is counterproductive in low traffic scenarios. Up to about 20 flows aggregation (both static and adaptive) have inferior delay and jitter values than no aggregation. With low traffic some packets are delayed due to the T_{delay} parameter, others can be processed immediately because enough packets are in the queue. As a result the packets require different times to be transferred over the network and the jitter increases. If T_{delay} is very low in this scenario, aggregation will show similar results as if no aggregation is done. However, for medium traffic scenarios T_{delay} is important to have a higher aggregation ratio.

Figures displaying the minimum, the maximum and 95% confidence interval for static aggregation and no aggregation can be found in the Appendix. The figures show, that if no aggregation is used the quality is more evenly distributed over the single flows (smaller confidence intervals).

5.2.2.3 Packet loss and delay in a temporal perspective

The average packet loss ratio or delay of one flow does not give yet enough information to assess the perceived quality. Jitter buffer and packet loss concealment only work with short time delay variations and bursty packet loss. Therefore it is also important to investigate the packet loss and delay for short intervals.

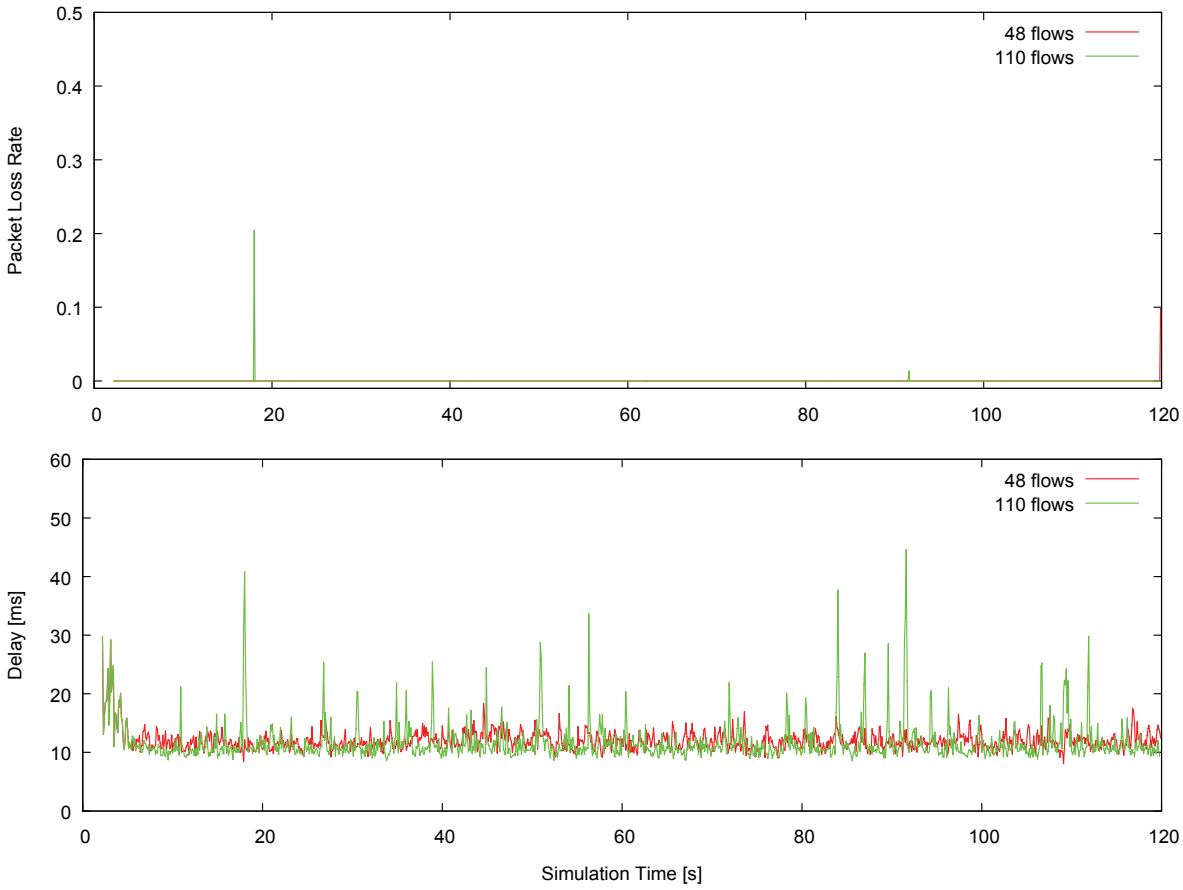

Figure 5.5 Packet loss and delay on the simulation time, adaptive aggregation

Figure 5.5 shows packet loss and delay for the whole simulation time of 120 seconds. The figure was obtained by calculating the average packet loss and delay of all flows for time intervals of 100 ms. For a stable network condition (48 flows) the packet loss ratio is almost zero and the delay is evenly distributed. It can be conducted, that the perceived speech quality is also very good in this condition. For 110 flows (which is close to the capacity threshold), the network shows signs of instability, which are represented by the delay spikes (i.e. higher jitter).

5.2.2.4 Aggregation efficiency

To study the efficiency of the aggregation algorithm channel utilization was analyzed. With a ns-2 extension[71] the time the channel is busy for a node due to the activity of other stations in its vicinity was measured.

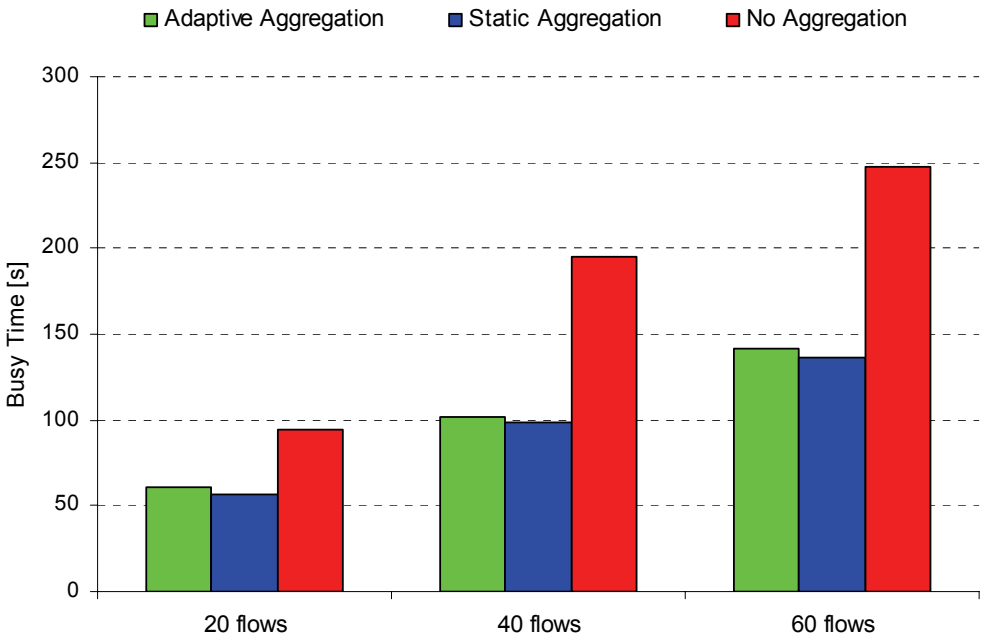

Figure 5.6 Channel Busy Times

Figure 5.6 shows the sum of busy times for the Nodes 2, 3, 4 and 5 for 20, 40 and 60 flows. Note that the total busy time can be greater than the simulation length, since a channel can be sensed busy by several stations at the same time. The diagram shows, that the channel is busy more often when no packet aggregation is done. Adaptive aggregation is 36%, 48% and 43% better than no aggregation. Static aggregation is slightly better than adaptive aggregation.

Another aspect of aggregation efficiency is how many packets are aggregated. Table 5.2 and Table 5.3 show aggregation statistics for the wireless nodes for 20 and 115 flows with adaptive aggregation. To also see how the algorithm aggregates packets a more detailed analysis of the cause of aggregation is done. The tables show the following values:

- Node: Node number in the Arrow Topology
- Number of aggregations by $SIZE_{min}$: number of aggregation operations which were performed due to the $SIZE_{min}$ threshold (left branch in Figure 4.6)
- Number of packets aggregated by $SIZE_{min}$: number of packets which were aggregated due to the $SIZE_{min}$ threshold
- Number of aggregations by T_{delay}: number of aggregation operations which were performed due to the T_{delay} threshold (right branch in Figure 4.6)
- Number of aggregated packets by T_{delay}: number of packets which were aggregated due to the T_{delay} threshold

- Total Aggregations: total number of aggregation operations i.e. number of generated IP_META packets
- Total Aggregated Packets: total number of packets which were aggregated. Total number of aggregated packets divided by total number of aggregations
- Average. Number aggregated Packets: average number of packets in one IP_META packet
- Number of packets sent without aggregation

For 115 flows the average number of aggregated packets is higher than for 20 flows. The intermediate nodes Node 2 and 3 have a higher aggregation ratio than the source nodes Node 4 and 5. That means that some nodes can aggregate more packets than others because more traffic is processed by them. In this case hop-by-hop aggregation can aggregate better than end-to-end aggregation. Still, the average number of aggregated packets is quite low. However, even the aggregation of only two packets means a large reduction of MAC layer overhead.

Since the T_{delay} threshold was chosen quite low, some packets are still sent without aggregation. For a higher network load this number decreases since the $SIZE_{min}$ threshold comes into effect most of the time.

Node	Number of aggregations by $SIZE_{min}$	Number of packets aggregated by $SIZE_{min}$	Number of aggregations by T_{delay}	Number of aggregated packets by T_{delay}	Total Aggregations	Total Aggregated Packets	Avg. Number aggregated Packets	Number of packets sent without aggregation
Node 2	2207	8828	2807	5620	5014	14448	2.88	6029
Node 3	8890	21143	4082	9119	12972	30262	2.33	10754
Node 4	5238	10478	0	0	5238	10478	2.00	33
Node 5	4996	9992	0	0	4996	9992	2.00	38

Table 5.2 Aggregation statistics for 20 flows, Adaptive Aggregation

Node	Number of aggregations by $SIZE_{min}$	Number of packets aggregated by $SIZE_{min}$	Number of aggregations by T_{delay}	Number of aggregated packets by T_{delay}	Total Aggregations	Total Aggregated Packets	Avg. Number aggregated Packets	Number of packets sent without aggregation
Node 2	26669	113678	230	461	26899	114139	4.24	1184
Node 3	58441	213230	4342	9766	62783	222996	3.55	5082
Node 4	26243	56308	0	0	26243	56308	2.15	24
Node 5	26784	57556	0	0	26784	57556	2.15	22

Table 5.3 Aggregation statistics for 115 flows, Adaptive Aggregation

5.2.3 Simulation 2 – Benefits of Adaptive Aggregation

The adaptive packet aggregation should be especially effective if a route consists of links with different channel quality. On a good link the $SIZE_{max}$ parameter can be higher than on a bad

link. Therefore more packets can be aggregated on a good link, while the aggregation ratio might be lower on a bad link. To verify this assumption the distance parameter d_0 and d_1 were varied and the performance of static and adaptive aggregation was studied.

Simulation Number	d_0	d_1	d_0-d_1	VoIP Capacity Static Aggregation	VoIP Capacity Adaptive Aggregation	Improvement Factor
3.1	56 m	56 m	0 m	130	112	0.86
3.2	56 m	55 m	1 m	68	115	1.69
3.3	60 m	55 m	5 m	56	108	1.93
3.4	65 m	55 m	10 m	32	70	**2.18**
3.5	65 m	50 m	15 m	26	32	1.23

Table 5.4 VoIP capacity with different channel qualities

Table 5.4 shows the number of supported flows for static aggregation and adaptive aggregation for different distance parameters. Simulations 3.1-3.4 show that a higher difference in the channel qualities (i.e. a higher difference between d_0 and d_1) yields a higher improvement due to the adaptive aggregation. *In simulation 3.4 the performance of adaptive aggregation is more than twice the performance of static aggregation*. With equal channel conditions the static aggregation is even slightly better (improvement factor = 0.86) because larger packets are created.

However, in simulation 5 the improvement factor drops again. The VoIP capacity of this topology is low. 32 respectively 26 flows do not produce enough traffic, for creating large aggregated packets. Therefore, also the static aggregation does not create packets, which are too large for the bad channel condition.

5.2.3.1 $SIZE_{max}$ and average packet size

Variation in the node distance should cause variation in the $SIZE_{max}$ parameter. To verify this, the average $SIZE_{max}$ for two links and different node distances was analyzed with 32 concurrent flows.

Simulation Number	d_0	d_1	$SIZE_{max}$ Node 4 → Node 3 [bytes]	Avg. IP_META size on link 4 → 3 [bytes]	$SIZE_{max}$ Node 2 → Node 3 [bytes]	Avg. IP_META size on link 2 → 3 [bytes]
3.1	56 m	56 m	993	141.13	294	112.75
3.2	56 m	55 m	1096	140.98	377	114.97
3.3	60 m	55 m	1144	142.12	396	116.88
3.4	65 m	55 m	171	146.93	507	114.63
3.5	65 m	50 m	145	145.23	745	114.78

Table 5.5 $SIZE_{max}$ for different distances

Table 5.5 shows the relation between the average $SIZE_{max}$ and the node distance. For example, as node distance d_0 changes from 56 m to 65 m, $SIZE_{max}$ drops from 1096 bytes to 171 bytes. Nevertheless, for same node distances (e.g. d_0 and d_1 in Simulation 1) $SIZE_{max}$ is

not necessarily the same. Therefore, it is important to measure the SNR and calculate the maximum packet size *at the receiver*, as it is done in the Simple Packet Size Selection Protocol.

Different values of $SIZE_{max}$ even for same node distances are caused by two factors:

1. $SIZE_{max}$ is a function of the SNR. The SNR is dependent on the receiving strength of the signal. In the shadowing propagation model the receiving strength for a fixed distance varies due to shadowing.
2. The SNR is also dependent on the background noise, i.e. also on parallel transmissions within the communications range.

Since even quite small variations in the SNR cause large changes in $SIZE_{max}$, $SIZE_{max}$ is quite volatile. For 42 flows $SIZE_{max}$ has hardly any influence on the average size of IP_META packets. In the simulation `sizefactor_` was set to 2.0, which means that the actual $SIZE_{max}$, which is used in the aggregation algorithm, is twice the value calculated and given in the table.

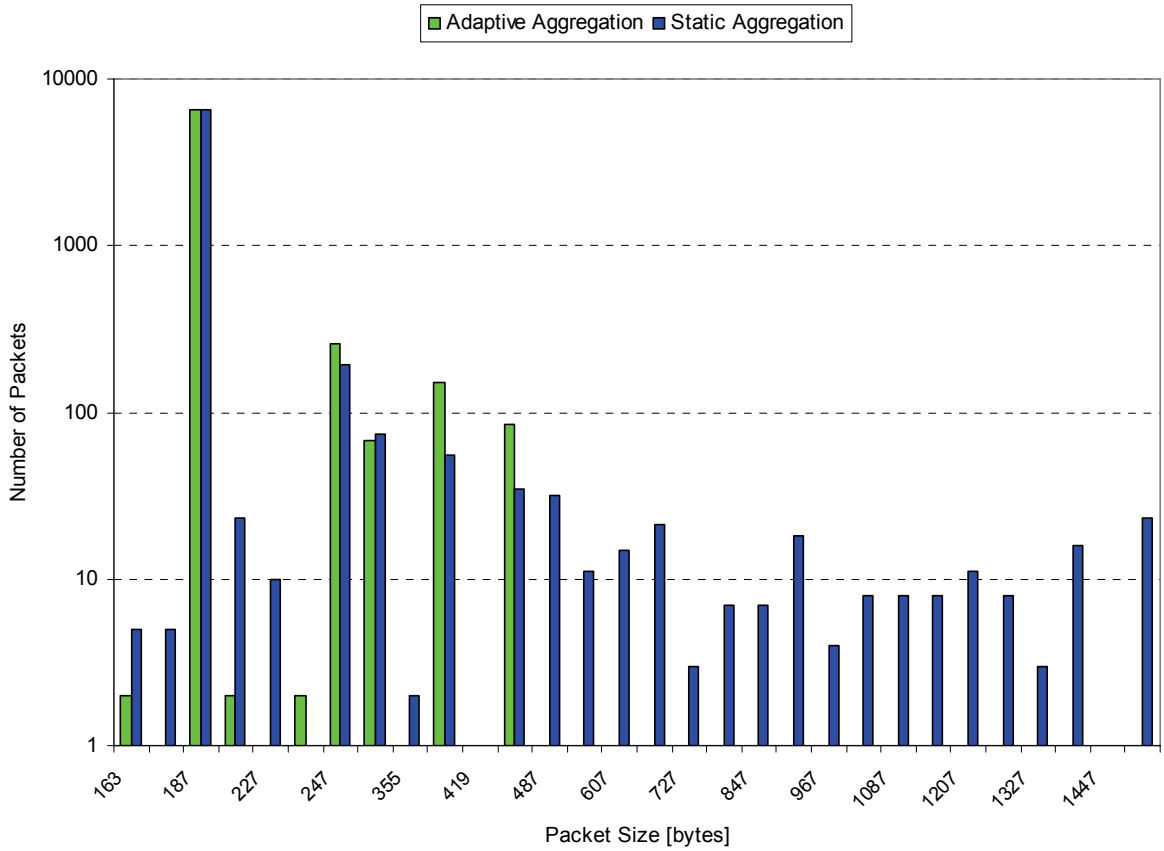

Figure 5.7 IP_META size distribution on link 4 → 3, d_0=65 m, d_1=55 m, 50 flows

Figure 5.7 shows the packet sizes of IP_META packet on the link between node 4 and 3 (mind the logarithmic scale on the Y-axis). With adaptive aggregation the packet size does not exceed 487 bytes, whereas static aggregation also produces many larger packets, which cannot be handled very well on the weak link. With adaptive aggregation IP_META packets are (re)-sent 1.62 times on MAC layer, with static aggregation 1.67 times (on link 4→3). This slightly higher value also indicates that the static aggregation uses too large packets.

5.2.4 Simulation 3 – Capacity with more hops

In the next simulation the VoIP capacity for two, three and four hops was determined. The original topology had two hops, by adding hops with a distance d_1 between Node 2 and Node 3 the hop-count was increased.

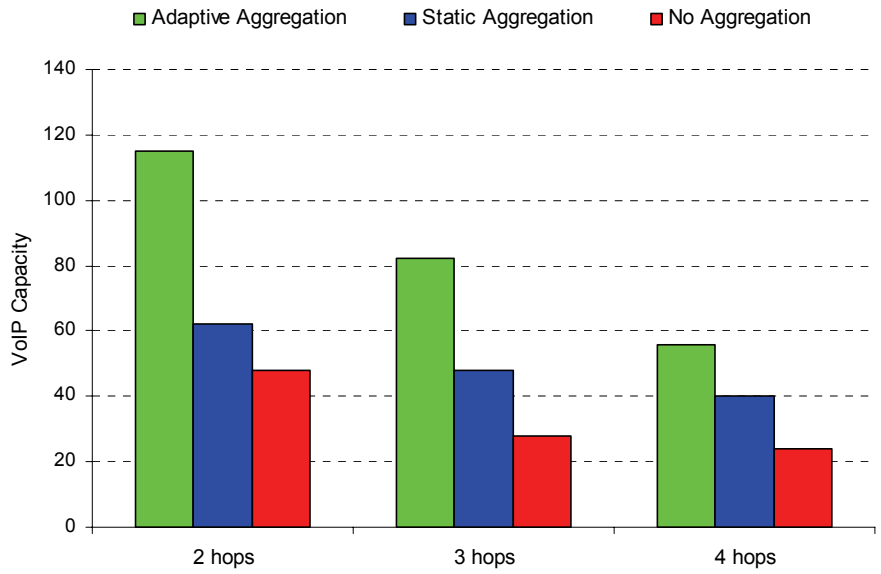

Figure 5.8 Number of supported flows for 2,3 and 4 wireless hops

Figure 5.8 shows the capacity for no aggregation, static aggregation and adaptive aggregation. It decreases strongly with the number of hops. This is mainly because the end-to-end packet loss ratio increases exponentially with the number of hops. In return the number of supported flows decreases with an factor of about $\frac{1}{n}$ for n hops[51].

5.2.5 Simulation 4 – Performance Tuning

The final simulations with the Arrow-topology should determine the effects of certain configuration settings in the MAC layer and the aggregation queue on the VoIP capacity.

5.2.5.1 Effects of RTS/CTS

The results in [15] indicate that RTS/CTS is inefficient if the interference range is not fully covered with RTS/CTS. [57] shows that disabling RTS/CTS is beneficial in combination with VoIP and adaptive packet aggregation. To see the effects of RTS/CTS on the VoIP capacity the number of supported flows with different RTS/CTS settings was determined (the other simulation settings are equal to Table 5.1).

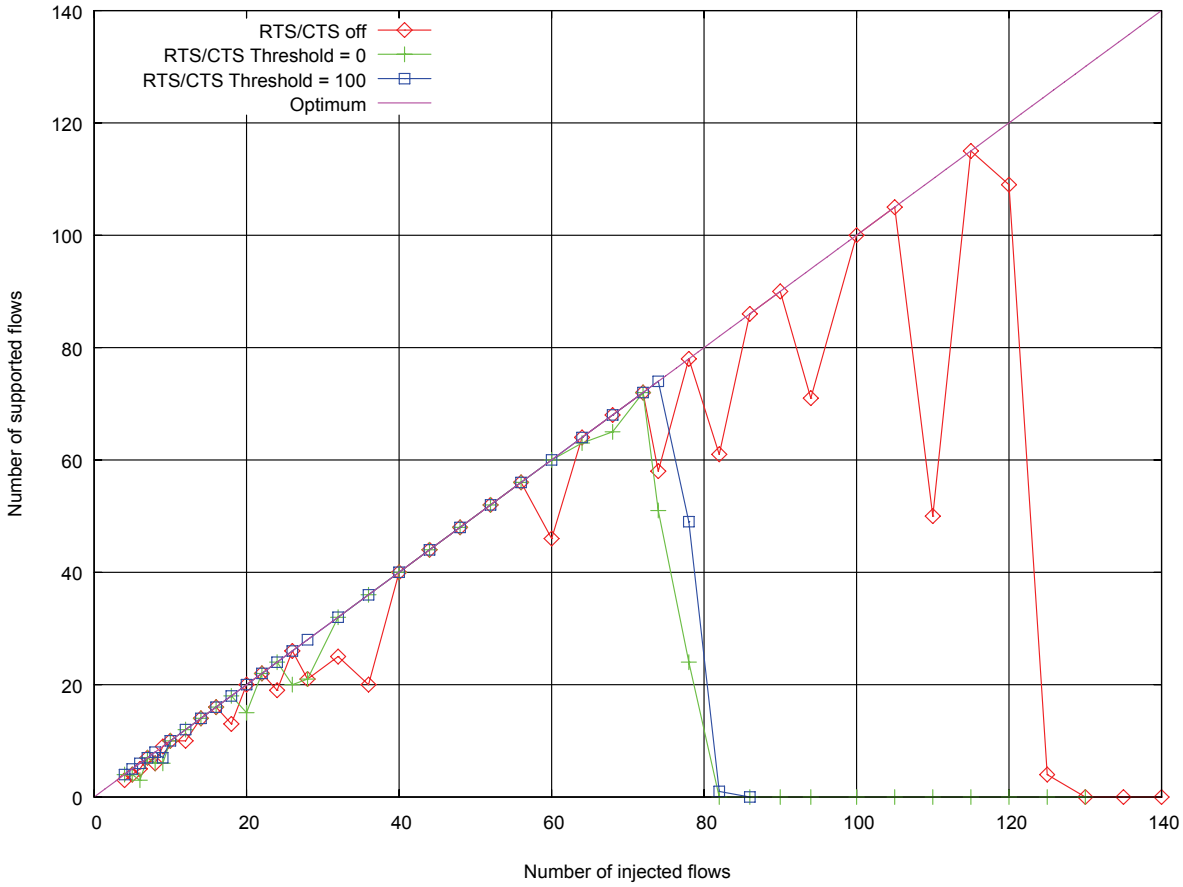

Figure 5.9 Supported flows with different RTS/CTS settings

Figure 5.9 shows the number of supported flows when RTS/CTS is off, is always used (RTS/CTS threshold = 0 bytes) and only used for aggregated packets (RTS/CTS threshold = 100 bytes, $SIZE_{min}$ = 101 bytes). The best performance can be achieved if RTS/CTS are disabled. If an RTS/CTS handshake is only performed for packets larger than 100 bytes, the VoIP capacity decreases from 115 to 74. If RTS/CTS are used for all packets only 72 flows are supported due to the high overhead of RTS/CTS. A positive aspect of using RTS/CTS is that the network is more stable and predictable. If the RTS/CTS threshold is set to 100 bytes,

up to the capacity threshold all injected flows are equally well supported. Therefore, it might be advantageous to enable RTS/CTS, although the overall capacity is lower.

5.2.5.2 sizefactor_, $SIZE_{min}$, T_{delay} and followup_

Preliminary simulations have shown that the calculation of $SIZE_{max}$ yields slightly too small sizes. Consequently an correction factor - the sizefactor_ - has been added to the aggregation queue (see section 4.2.2.1). In the previous simulations the sizefactor_ was set to 2.0. To evaluate the impact of the size factor, the VoIP capacity for different node distances and size factors has been analyzed.

Simulation Number	d_0	d_1	sizefactor_	VoIP Capacity
4.1	56 m	55 m	1.5	105
4.2	56 m	55 m	2.0	115
4.3	56 m	55 m	2.2	120
4.4	56 m	55 m	2.5	130
4.5	60 m	55 m	1.8	90
4.6	60 m	55 m	2.5	110
4.7	65 m	55 m	1.8	68
4.8	65 m	55 m	2.5	28

Table 5.6 Influence of sizefactor_ on VoIP capacity

Table 5.6 shows that a sizefactor_ of 2.5 is best for node distances of 56 and 60 meters (Simulations 4.1-4.6). For 65 meters 1.8 gives the better results (Simulations 4.7 and 4.8). That means that the factor by which the SNR-packet size calculation calculates a too small packet size is not linear for different link qualities. It can therefore be concluded, that the present $SIZE_{max}$ calculation does not work optimal and that the sizefactor_ improves the performance. A more accurate calculation which is solely based on the SNR (e.g. by increasing the allowed packet loss due to FER) and some other metrics (e.g. queue size) and which works for all distances should be found.

Simulation Number	$SIZE_{min}$	VoIP capacity
4.9	101 bytes	115
4.10	150 bytes	130
4.11	200 bytes	125

Table 5.7 Influence of $SIZE_{min}$ on the VoIP capacity

Beside the $SIZE_{max}$ the $SIZE_{min}$ parameter is a performance factor. Table 5.7 shows that a $SIZE_{min}$ of 150 bytes gives the best capacity. Setting $SIZE_{min}$ dynamically as a multiple of $SIZE_{max}$ (e.g. $SIZE_{\min} = 0.5 * SIZE_{\max}$) might be an approach that should be investigated further.

The T_{delay} setting only has a minor influence on the VoIP capacity. When enough traffic is generated, almost all packets are aggregated due to the $SIZE_{min}$ threshold. Since the T_{delay} does only play a role in low traffic scenarios, also the followup_ parameter is not significant for maximizing throughput.

5.3 Grid Topology

The second topology used is an 8x8 grid, as illustrated in Figure 5.10. The distance between two neighbouring nodes is 55 m. The topology only comprises static, wireless nodes, which can generate, aggregate, de-aggregate and forward traffic. In this topology the ETX is used as a routing metric since it produces more stable and better routes than the minimum hop-count metric. The ETX needs a few seconds to be settled. Therefore in the first 30 seconds of the simulation no traffic is generated. Afterwards 60 seconds of VoIP communications are simulated.

5.3.1 Random generation of flows

The source and destination nodes of the RTP-streams are selected pseudo-randomly. The source is chosen completely randomly among all nodes within the grid. To control the average hop-count a destination node in the vicinity of the source is selected. This is done by randomly hop *x* nodes left or right to the source and *y* nodes up or down. For example if the source is Node 17, *x* is 2 and *y* is -2 the destination will be Node 3. The maximum possible values of *x* and *y* therefore control the hop count. In performed simulations the random interval was set to [-3, 3].

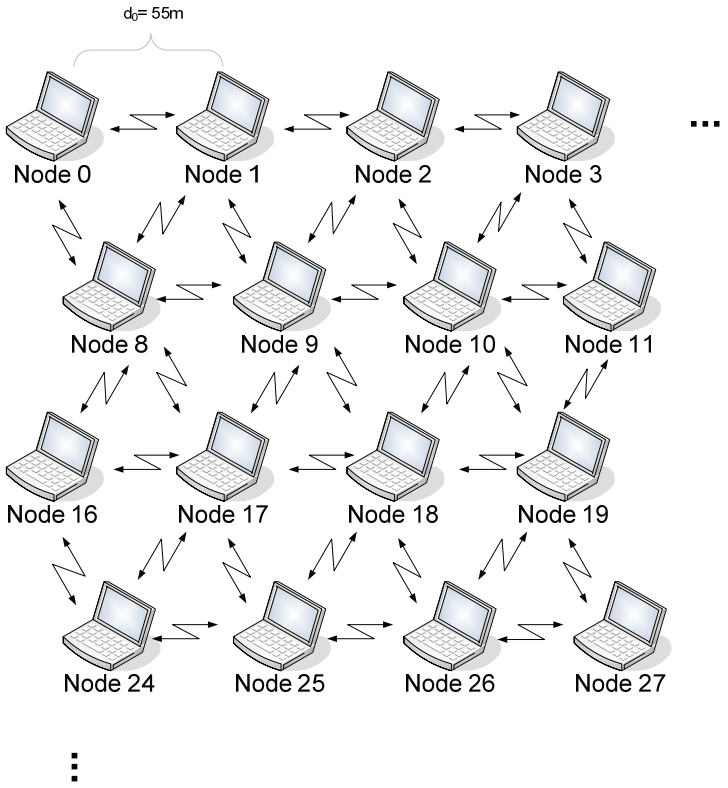

Figure 5.10 Grid Topology

5.3.2 Simulation 1 – General Performance Analysis

Figure 5.11 shows the number of supported flows for no aggregation, static aggregation and dynamic aggregation. The results are the average of five simulation runs with pseudo-random sender-receiver configurations. For all aggregation modes the sender-receiver pairs were the same. The number of injected flows was gradually increased from 4 to 160. The number of supported flows was analyzed for every traffic setting.

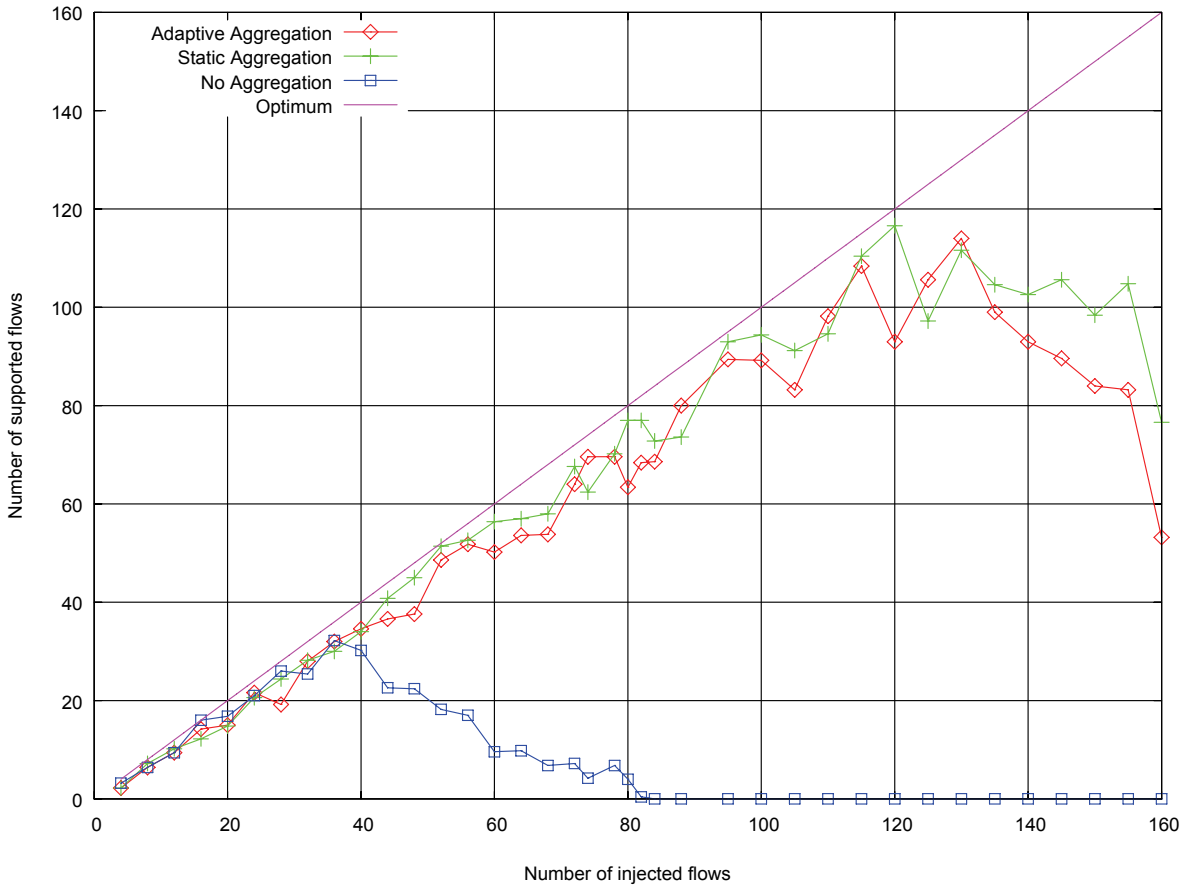

Figure 5.11 Supported flows

Again the simulation results in Figure 5.11 show, that the network has some immanent capacity threshold. However what and where this threshold is, is not as obvious as in the Arrow Topology. Even in low traffic settings there are always some flows, which are not supported. Due to the random source-destination selection the path length and particularly the routing stability cannot be controlled as nicely as in the arrow topology. In addition, the random traffic generation might not distribute the traffic load completely even over the whole grid.

Consequently, the behaviour that (almost) all flows are supported and beyond the capacity threshold the number of flows drops to zero immediately cannot be observed in a grid. Therefore, a more reasonable definition of *VoIP capacity* in the grid is the point at which additionally added flows do not result in more supported flows. With aggregation this point is at about 120 flows, for no aggregation at about 35 flows. Thus, aggregation more than triples the capacity.

5.3.3 Simulation 2 – Different Node Distances

In the last simulation the grid nodes had different distances. Nodes within one row (e.g. Node 0 and 1) had a distance of 55 m. The distances between nodes of different two rows had a distance of 57 m (e.g. Node 0 and 8) and 54 m (e.g. Node 1 and 8).

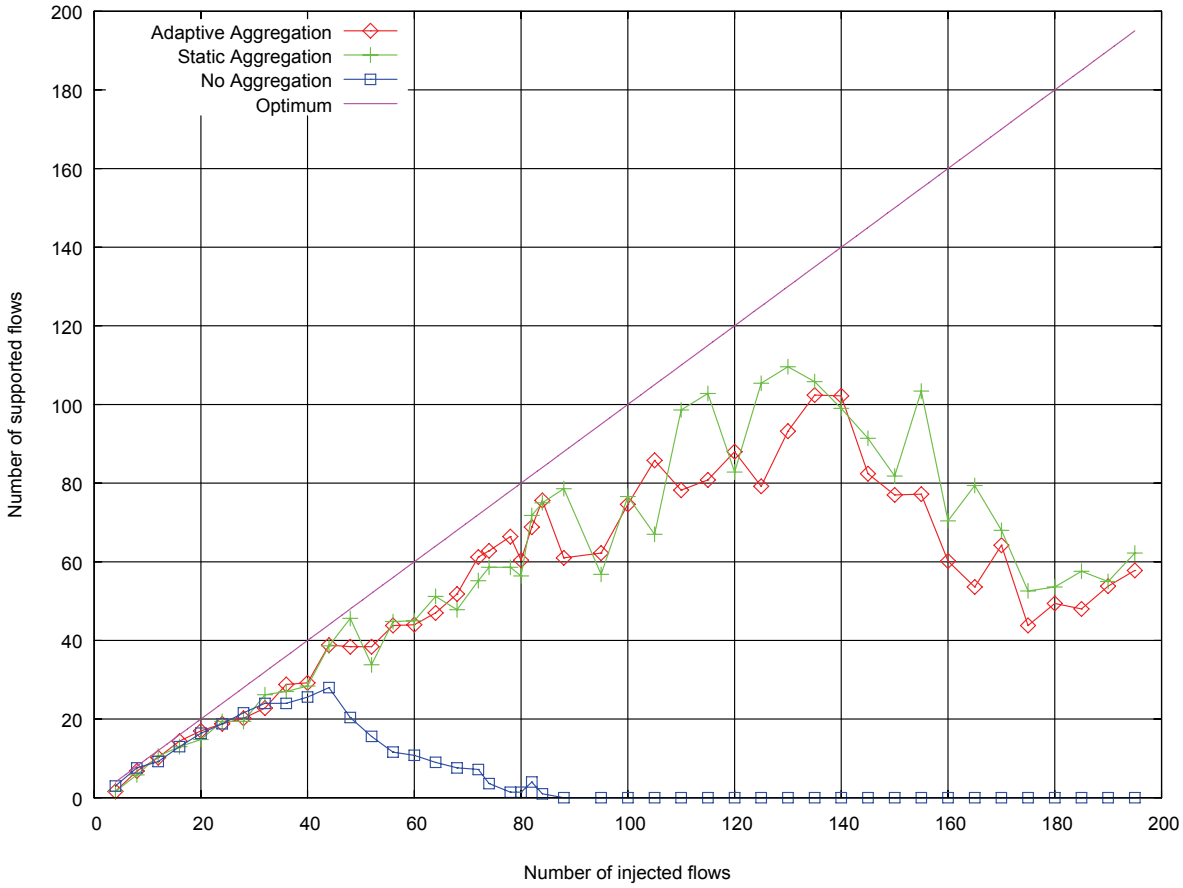

Figure 5.12 Supported Flows with different node distances

Figure 5.12 shows that adaptive aggregation and static aggregation perform almost equally well. The benefit of adaptive aggregation cannot be seen in this simulation since the routing might omit routes with bad link quality.

6 Conclusion

6.1 Evaluation of Results and Accomplishments

Packet Aggregation can improve the performance of Voice over IP in Wireless Mesh Networks significantly. Depending on the network topology packet aggregation can increase the VoIP capacity by more than 300%. In low traffic scenarios packet aggregation should not be used. Adaptive aggregation is beneficial when the link quality is low and different link qualities are involved. Adaptive aggregation requires a mechanism to determine the optimum packet size. The packet size calculation needs to be based on several metrics, which cover the different reasons for packet loss. Some metrics can only be obtained by the receiver. Therefore a protocol for calculating the size is necessary. A novel aggregation scheme and two protocols for determining the packet size were proposed. The aggregation scheme and the Simple Packet Size Selection Protocol were implemented in ns-2 and evaluated.

6.2 Problems and Solutions

At this point, the two most severe problems, which caused the most delay and needed the most effort to be resolved, will be mentioned.

6.2.1 Wrong Theoretical Approach

The initial approach, to select the packet size based on the ETX turned out to be non applicable. As argued in section 3.2.3.1 the concept should not give meaningful results. This assumption could also be verified in simulations. The problem was solved by the Simple and Enhanced Packet Size Selection Protocol. Still the question remains open, why the very similar IPAC method works.

6.2.2 Route Stability

One of the most time consuming problems was the creation of a stable topology. While it is easy to create stable routes when the two-ray ground model is used, it is very tricky to achieve the same result with the shadowing model and the FER-patch. The problem details and the solution were described in section 5.1. It should be noted, that the instable routes occurred in ns-2 due to the applied patches, but could also arise in a real test-bed.

A second factor that caused route instability was that due to the FER-path a rather large amount of AODV-HELLO messages was lost. Usually a node invalidates a route if three consecutive AODV-HELLO messages are not received. By increasing this threshold to 5 the routes were stable. An additional problem was route instability due to packet loss. To minimize this problem, the β-patch was applied to AODV-UU.

6.3 Future work

The work on the master's thesis raised a lot of interesting questions and issues, which could not be addressed due to time constraints. Some suggestions for future work in this field are:

6.3.1 Evaluate perceived Voice Quality

[72] provides a patch for ns-2 that allows to measure the perceived end-to-end voice transmission quality within ns-2. It includes an own agent for generating VoIP traffic and a number of programs that calculate the voice quality based on the trace files. Using PESQ or the E-Model an objective quality metric can be calculated. This method promises a more accurate prediction of the voice quality than the analysis in Chapter 5. Therefore, it would be interesting to evaluate the packet aggregation with this model.

6.3.2 Improve Aggregation Algorithm

Currently packets are always forwarded, even though a receiver might not be able to use them, because their delay or jitter is too high. To reduce resource usage by "useless" packets, intermediate nodes should only forward packets, when the delay and jitter is below a certain threshold.

The use of header compression promises an remarkable increase of VoIP capacity[51]. A new header compression algorithm must work with inter-flow and intra-flow aggregation. The efficiency of header compression should be evaluated.

6.3.3 Implement Enhanced Packet Size Selection Protocol

The Enhanced Packet Size Selection Protocol needs to be implemented and evaluated in ns-2. A simulation can show if the theoretical approach is right and how good it performs.

6.3.4 Call Admission Control/Prioritization/Resource Reservation

The results in chapter 4 show, that for a given topology a capacity threshold exists. If only one or two flows are added beyond this threshold no flow achieves good quality (at least in the Arrow Topology). Future research should find out, how this capacity threshold can be

determined and how a distributed call admissions scheme can be implemented. The call admission should limit the number of allowed calls to the network capacity. An alternative would be to allow an arbitrary number of calls, but providing enough resources only to a number which is smaller than the capacity. This can be done either with traffic prioritization, resource reservation or a combination of both.

6.3.5 Personal Conclusion

The work on this thesis was very interesting and challenging. It started as a small project for the "Topics in Computer Networking" course and ended in hours of discussions with my adviser, thousands of simulations, hundreds of lines of code, dozens diagrams and, I hope, in an interesting thesis paper. In the past six months I learnt a lot about ns-2, C++ and Perl programming, WMNs and Voice over IP. I believe that I achieved all goals which I proposed in Chapter 1. However, many questions are still open and should be answered in future projects.

References

[1] Murthy, C.S.R. and Manoj, B.S., *Ad Hoc wireless networks : architectures and protocols*. Low Price Edition ed. Prentice Hall communications engineering and emerging technologies series. 2004, Upper Saddle River, NJ: Prentice Hall PTR, xxii, 857 p., ISBN: 013147023X.

[2] Conner, W.S., et al., *IEEE 802.11s Tutorial - Overview of the Amendment for Wireless Local Area Mesh Networking*. 2006.

[3] Bicket, J., et al. *Architecture and Evaluation of an Unplanned 802.11b Mesh Network*. in Proceedings of *Mobicom 2005*. 2005. Colonge, Germany.

[4] *Iceland Embraces Wireless Mesh for Voice and Data Connectivity in Remote Communities*. PR Newswire, 2007 [cited 23.5.2007]; URL: http://sev.prnewswire.com/computer-electronics/20070523/AQW30223052007-1.html.

[5] The VINT Project, *The ns Manual*, K. Fall and K. Varadhan, Editors. 2007, UC Berkeley, LBL, USC/ISI and Xerox PARC.

[6] Ohrtman, F., *Voice over 802.11* 2004, Boston Artech House, ISBN: 1580536786.

[7] Faccin, S.M., et al., *Mesh WLAN networks: concept and system design.* in *IEEE Wireless Communications*, 2006. 13(2): p. 10-17.

[8] Institute of Electrical and Electronics Engineers, *Joint SEE-Mesh/Wi-Mesh Proposal to 802.11 TG*. 2006, Institute of Electrical and Electronics Engineers,.

[9] Akyildiz, I.F., Wang, X., and Wang, W., *Wireless mesh networks: a survey.* in *Computer Networks*, 2005. 47(4): p. 445-487.

[10] Schiller, J.H., *Mobile communications* 2nd ed. 2003: Addison-Wesley, ISBN: 0-321-12381-6.

[11] Wheat, J., et al., *Designing a wireless network*. 2001, Rockland, Mass.: Syngress Publishing, 409, ISBN: 1928994458.

[12] Kuran, M.S. and Tugcu, T., *A survey on emerging broadband wireless access technologies.* in *Computer Networks*, 2006. In Press, Corrected Proof.

[13] Chandra, A., Gummalla, V., and Limb, J.O., *Wireless Medium Access Control Protocols.* in *IEEE Communications Surveys*, 2000. 3(2): p. 2–15.

[14] Kumar, S., Raghavan, V.S., and Deng, J., *Medium Access Control protocols for ad hoc wireless networks: a survey*, E. B.V., Editor. 2004.

[15] Chatzimisios, P., Boucouvalas, A.C., and Vitsas, V., *Effectiveness of RTS/CTS handshake in IEEE 802.11 a Wireless LANs.* in *Ad Hoc Network Journal*, 2003. 1(1): p. 107-123.

[16] Tzu-Jane, T. and Ju-Wei, C. *IEEE 802.11 MAC protocol over wireless mesh networks: problems and perspectives*. in Proceedings of *19th International Conference on Advanced Information Networking and Applications*. 2005. Taipei, Taiwan: IEEE Comput. Soc.

[17] Xu, S. and Saadawi, T., *Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?* in *IEEE Communications Magazine*, 2001. 39.

[18] Ho Ting Cheng, H.J.W.Z., *Distributed medium access control for wireless mesh networks.* in *Wireless Communications and mobile computing*, 2006. 6(6): p. 845-864.

[19] Feit, S., *TCP/IP: architcture, protocols, and implementation with IPv6 and IP security* 1999, New York: McGraw-Hill, ISBN: 0070220697

[20] Stevens, W.R., *TCP/IP Illustrated*. Vol. 1: The Protocols. 1994: Addison-Wesley, ISBN: 0-201-63346-9.

[21] Yang, Y., Wang, J., and Kravets, R. *Designing Routing Metrics for Mesh Networks*. in Proceedings of *First IEEE Workshop on Wireless Mesh Networks*. 2005. Santa Clara, CA: IEEE.

[22] Richard, D., Jitendra, P., and Brian, Z. *Comparison of routing metrics for static multi-hop wireless networks*. in Proceedings of *Conference on Applications, technologies, architectures, and protocols for computer communications*. 2004. Portland, Oregon, USA: ACM Press.

[23] De Couto, D.S.J., et al., *A high-throughput path metric for multi-hop wireless routing.* in *Wireless Networks*, 2005. 11(4): p. 419-34.

[24] Waharte, S., et al., *Routing protocols in wireless mesh networks: Challenges and design considerations.* in *Multimedia Tools and Applications*, 2006. 29(3): p. 285-303.

[25] Richard, D., Jitendra, P., and Brian, Z. *Routing in multi-radio, multi-hop wireless mesh networks*. in Proceedings of *10th annual international conference on Mobile computing and networking*. 2004. Philadelphia, PA, USA: ACM Press.

[26] Perkins, C., Belding-Royer, E., and Das, S., *Ad hoc On-Demand Distance Vector (AODV) Routing, RCF 3561*. 2003, Internet Engineering Task Force.

[27] Institute of Electrical and Electronics Engineers, *802.11s Proposal - Joint SEE-Mesh/Wi-Mesh Proposal to 802.11 TGs*, in *IEEE 802.11-06/0328r0*. 2006, IEEE.

[28] Abolhasan, M., Wysocki, T., and Dutkiewicz, E., *A review of routing protocols for mobile ad hoc networks.* in *Ad Hoc Networks*, 2004. 2(1): p. 1-22.

[29] *freifunk.net website*. 2007 [cited 8.3.2007]; URL: http://freifunk.net/.

[30] Stafford, M., *Signaling and switching for packet telephony* 2004, Boston: Artech House, ISBN: 1580537375.

[31] Cox, R.V., *Three new speech coders from the ITU cover a range of applications.* in *IEEE Communications Magazine*, 1997. 35(9).

[32] Telecommunications Industry Association, *TSB-116-A - Voice Quality Recommendations for IP Telephony*, in *TIA Telecommunications Systems Bulletin*. 2006, Telecommunications Industry Association: Arlington, USA.

[33] Schulzrinne, H. and Casner, S., *RTP Profile for Audio and Video Conferences with Minimal Control, RFC 3551*. 2003, Internet Engineering Task Force.

[34] International Telecommunication Union, *Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s, G.723.1*, in *General Aspects of of Digital Transmission Systems*, I.T. Union, Editor. 2006.

[35] International Telecommunication Union, *Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP), G.729*, in *General Aspects of of Digital Transmission Systems*, I.T. Union, Editor. 1996.

[36] Groom, F.M. and Groom, K.M., *The basics of voice over Internet protocol*. 2004, Chicago: International Engineering Consortium, ISBN: 0-931695-26-0.

[37] Harry, J., *Voice and Video Over IP* 2002: McGraw-Hill Professional, ISBN.

[38] Handley, M., et al., *SIP: Session Initiation Protocol, RFC 2543*. 1999, Internet Engineering Task Force.

[39] Schulzrinne, H., et al., *RTP: A Transport Protocol for Real-Time Applications, RFC 1889*. 1996, Internet Engineering Task Force.

[40] Bormann, C., et al., *RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed, RFC 3095*. 2001, Internet Engineering Task Force.

[41] International Telecommunication Union, *One-way transmission time, G.114*. 2003, International Telecommunication Union: Transmission Systems and Media, Digital Systems and Networks.

[42] Hardy, W.C., *VoIP service quality: measuring and evaluating packet-switched voice* 2003, New York: McGraw-Hill, ISBN: 0071410767.

[43] International Telecommunication Union, *Transmission impairments due to speech processing, G.113*, in *Transmission Systems and Media, Digital Systems and Networks*, International Telecommunication Union, Editor. 2001, International Telecommunication Union,.

[44] Sanneck, H., Carle, G., and Koodli, R. *Framework model for packet loss metrics based on loss runlengths*. in Proceedings of *SPIE/ACM SIGMM Multimedia Computing and Networking Conference 2000*. 2000. San Jose, CA: SPIE/ACM SIGMM.

[45] International Telecommunication Union, *Methods for subjective determination of transmission quality, P.800* in *Series P: Telephone Transmission Quality*. 1996, International Telecommunication Union.

[46] International Telecommunication Union, *The E-model, a computational model for use in transmission planning, G.107*, in *Transmission Systems and Media, Digital Systems and Networks*. 2005, International Telecommunication Union.

[47] International Telecommunication Union, *Methods for objective and subjective assessment of quality, P.861*, in *Series P: Telephone Tranmission Quality, Telephone Installations, Local Line Networks*. 1996, International Telecommunication Union.

[48] Niculescu, D., et al. *Performance of VoIP in a 802.11 Wireless Mesh Network*. in Proceedings of *IEEE INFOCOM 2006*. 2006.

[49] Ganguly, S., et al., *Performance Optimizations for Deploying VoIP Services in Mesh Networks*. 2006. p. 2147-2158.

[50] Kyungtae, K., et al. *On packet aggregation mechanisms for improving VoIP quality in mesh networks*. in Proceedings of *63rd Vehicular Technology Conference* 2006. Melbourne, Vic., Australia: IEEE.

[51] Kyungtae, K. and Sangjin, H. *VoMESH: voice over wireless mesh networks*. in Proceedings of *IEEE Wireless Communications and Networking Conference* 2006. Las Vegas, NV, USA: IEEE.

[52] Jain, A., et al., *Benefits of packet aggregation in ad-hoc wireless network*. 2003, Tech. Rep. CU-CS-960-03, University of Colorado at Boulder, 2003.

[53] Sharif, H. and Ci, S. *Adaptive Optimal Frame Length Predictor for IEEE 802.11 Wireless LAN*. in Proceedings of *6th International Symposium on Digital Signal Processing for Communication Systems* 2002.

[54] Chen, C.-c., et al. *Rate-adaptive Framing for Interfered Wireless Networks*. in Proceedings of *IEEE INFOCOM*. 2007. Anchorage, Alaska, USA.

[55] Lin, Y. and W.S. Wong, V. *Frame Aggregation and Optimal Frame Size Adaptation for IEEE 802.11n WLANs*. in Proceedings of *IEEE Global Telecommunications Conference*. 2006. San Francisco, CA.

[56] Raghavendra, R., et al. *IPAC – An IP-based Adaptive Packet Concatenation for Multihop Wireless Networks*. in Proceedings of *Asilomar Conference on Systems, Signals and Computing*. 2006. Pacific Grove, CA.

[57] Kassler, A.J. and Dely, P. *On Packet Aggregation for VoIP in Wireless Meshed Networks*. in Proceedings of *7th Scandinavian Workshop on Wireless Ad-hoc Networks*. 2007. Stockholm, Sweden.

[58] Lettieri, P. and Srivastava, M.B. *Adaptive Frame Length Control for Improving Wireless Link Throughput, Range and Energy Efficiency*. in Proceedings of *INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies*. 1998. San Francisco, USA: IEEE.

[59] Barsocchi, P., Oligeri, G., and Potortì, F., *Frame error model in rural Wi-Fi networks*. 2006.

[60] Mangold, S., Choi, S., and Esseling, N. *An Error Model for Radio Transmissions of Wireless LANs at 5GHz*. in Proceedings of *10th Aachen Symposium on Signal Theory*. 2001. Aachen.

[61] Xiuchao, W., *Simulate 802.11b channel within ns2*. 2004, National University of Singapore: Singapore.

[62] Wu, X. and Ananda, A.L. *Link characteristics estimation for IEEE 802.11 DCF based WLAN*. in Proceedings of *29th Annual IEEE International Conference on Local Computer Networks*. 2004. Tampa, FL, USA: IEEE (Comput. Soc.).

[63] Kim, J., et al. *CARA: Collision-Aware Rate Adaptation for IEEE 802.11 WLANs*. in Proceedings of *IEEE INFOCOM 2006*. 2006. Barcelona, Spain.

[64] NIST/SEMATECH. *e-Handbook of Statistical Methods*. 2006 [cited 5.5.2007]; URL: http://www.itl.nist.gov/div898/handbook/pmc/section4/pmc431.htm.

[65] Chung, J. and Claypool, M. *NS by Example*. [cited 17.04.2007]; URL: http://nile.wpi.edu/NS/.

[66] Wiberg, B., *Porting AODV-UU Implementation to ns-2 and Enabling Trace-based Simulation*, Master's Thesis, Uppsala University, 2002

[67] *NS-2 Trace Formats*. 2007 [cited 22.04.2007]; URL: http://nsnam.isi.edu/nsnam/index.php/NS-2_Trace_Formats.

[68] Karlstad University. *DISCO/MeshWiki/NS Resources*. 2007 [cited 22.5.2007]; URL: http://www.cs.kau.se/cs/prtp/pmwiki/pmwiki.php?n=MeshWikipage.NS2Resources.

[69] Nahm, K., Helmy, A., and Jay Kuo, C.-C. *Improving Stability and Performance of Multihop 802.11 Networks*. in Proceedings of *ACM MobiHoc Urbana-Champaign*. 2005. IL.

[70] International Telecommunication Union, *ITU-T Recommendation P.59 - Artificial conversational speech*. 1993, International Telecommunication Union.

[71] Garetto, M., Salonidis, T., and Knightly, E. *Modeling Per-flow Throughput And Capturing Starvation in CSMA Multi-hop Wireless Networks*. in Proceedings of *IEEE INFOCOM 2006*. 2006. Barcelona, Spain.

[72] Wolisz, A. *Assessing the Perceptual Quality of VoIP Transmissions*. 2007 [cited 25.5.2007]; URL: http://www.tkn.tu-berlin.de/research/qofis/.

A Appendix

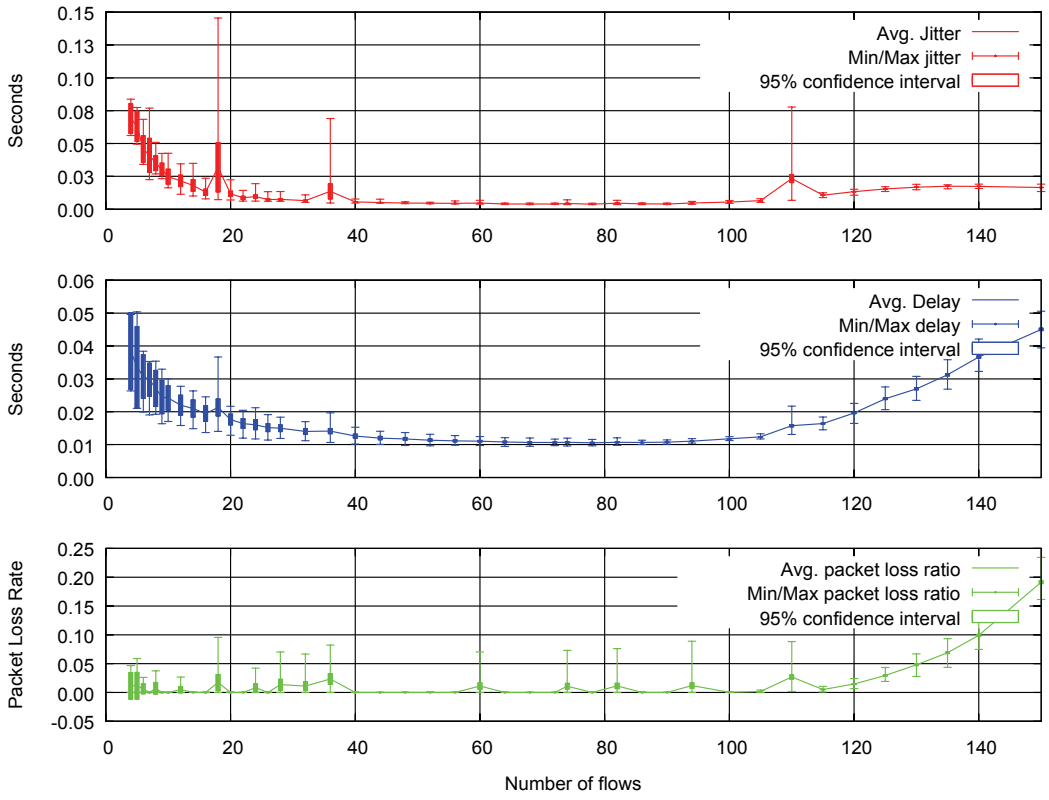

Figure A.0.1 Packet Loss, Delay and Jitter for Static Aggregation

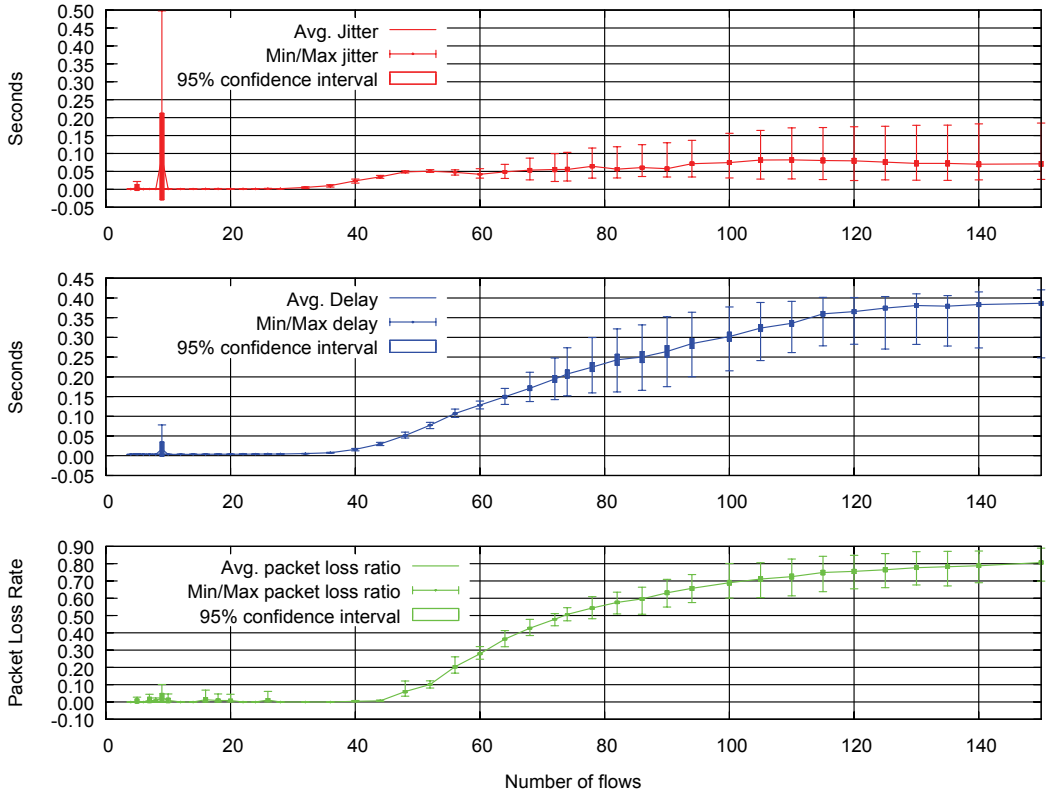

Figure A.0.2 Packet Loss, Delay and Jitter for No Aggregation

File	Description
adov-uu/aodv-uu.cc	Modified code
aodv-uu/aodv-hello.c	Added code for handling the exchanging packet size information
aodv-uu/debug.c	Modified code to
aodv-uu/defs.h	Added parameters for calculating packet size
aodv-uu/params.h	Added parameters for calculating packet size
aodv-uu/routing-table.h	Added fields for packet size and SNR in routing table
common/ip-meta.h/cc	Includes definition of IP_META packets
common/packet.h	Added definition for IP_META
common/packer.h/cc	Implements a helper-class for creating IP_META packets
Makefile.in	Added new files as target
queue/agent-deaggregator.h/cc	Implements Agent/Deaggregator
queue/aggregator-adaptive.h/cc	Implements Queue/Aggregator/Adaptive
tcl/lib/ns-defaults.tcl	Added default parameters for new OTcl objects
tcl/lib/ns-packet.tcl	Added definition for IP_META
trace/cmu-trace.cc	Added output for IP_META and flow-id
_thesis	Contains simulation scripts and analysis tools
mac/mac_802-11.c	Added FER calculation patch. Added code for monitoring SNR of incoming packets.
mac/ErrorModel80211.h/c	Code for calculating the SNR

Table A.1 List of modified/added files in ns-2 source code

The attached CD-ROM includes the modified ns-2 sources, the used simulation scripts and a VMware-Image, with an Mandrake Linux, some development tools and an executable version of ns-2. The image can be loaded with VMware-Player, which is freely available from www.vmware.com. The username and password for the Linux login is “ns”. The modified version of ns-2 can be found in “/home/ns/workspace”.