QoS Provisioning in WLAN Mesh Networks Using Dynamic Bandwidth Control

D. Hock*, N. Bayer†, R. Pries*, M. Siebert†, D. Staehle*, V. Rakocevic‡, B. Xu†
*Deptartment of Distributed Systems, University of Würzburg, Germany
Email: {hock, pries, staehle}@informatik.uni-wuerzburg.de
†Deutsche Telekom/T-Systems, Darmstadt, Germany
Email:{Nico.Bayer, M.Siebert, Bangnan.Xu}@t-systems.com
‡School of Engineering and Mathematical Sciences, City University, London, UK
Email:V.Rakocevic@city.ac.uk

***Abstract*—WLAN, based on the IEEE 802.11 standard has been extensively studied since its release. In addition to infrastructure access to WLAN, mesh networks currently attract a lot of attention. This comes from the envisioned advantages of wireless mesh networks, such as cheap installation costs, extended coverage, robustness, easy maintenance, and self-configuration possibilities. In this paper we focus on Quality of Service support for multimedia applications in WLAN-based mesh networks. Therefore, a dynamic bandwidth control mechanism is implemented on the network layer and the results show that high prioritized traffic can be protected from disturbing best effort traffic.**

***Index Terms*—WLAN, 802.11, Mesh, Testbed**

I. Introduction

The continuous standardization of *Wireless Local Area Networks* (WLANs) is a success story. Since the first release of the IEEE 802.11 WLAN standard in 1997, it gradually improved its performance and evolved into a very flexible and well-understood technology. However, todays WLANs are mainly *Access Point* (AP) centered and form small islands in laboratories, on campuses, and in hot-spot urban environments. A *Wireless Mesh Network* (WMN) brings these hot-spots together, similar to wired routers, which connect networks to ensure a reliable end-to-end connection. The standardization of WLAN mesh networks was started in 2003 under the extension IEEE 802.11s [1]. Besides the IEEE 802.11s standard further standardization groups for WMNs like IEEE 802.15.5 [2] and IEEE 802.16j [3] underline the importance of wireless mesh networks.

The main characteristic of a wireless mesh network is the communication between nodes over multiple wireless hops to increase the radio coverage and to enable network connectivity between stations which are outside their direct receive range. In contrast to wireless ad-hoc networks which focus on mobility, end user devices, and point to point connections, WMNs are normally static devices and focus on reliability, network capacity, and are mainly used as an alternative to a wired network infrastructure.

Major research aspects in WMNs are routing and *Quality of Service* (QoS) support. In this paper, we present a distributed, measurement-based approach to protect QoS traffic from best effort traffic in WLAN-based mesh networks. The aim of the proposed mechanism is to keep track of the services currently present in the network and to ensure a stable and high QoS level. The tools for the approach are implemented and tested on wireless mesh nodes. The results reveal that the mechanism does not only keep track of disturbing traffic on the same path, but also regulates traffic flows on crossing paths. It should be noted that the presented approach is general in nature and even if it is discussed and implemented based on wireless mesh networks, it can also be applied to other types of networks, e.g. wireless ad-hoc and wired networks.

The remainder of the paper is organized as follows. In Section II the work related to QoS issues in wireless mesh networks is shown. This is followed by Section III, introducing wireless mesh networks and its known problems. Our approach is presented in Section IV and Section V shows the results of performance measurements in an example scenario. Finally, a short conclusion is given in Section VI.

II. Related Work

One step towards QoS support in IEEE 802.11 networks is defined in the IEEE 802.11e standard for service differentiation, which slightly modifies the *Carrier Sense Multiple Access/Collision Avoidance* (CSMA/CA) mechanism. However, the standard does not guarantee a good QoS level, especially in highly loaded networks. This has been tested and improved for single hop environments in [4], [5], and [6].

A MAC protocol for QoS support in WMNs is proposed by Carlson et al. [7]. It is called *Distributed end-to-end Allocation of time slots for REal-time traffic* (DARE). In this protocol, time slots are reserved in all mesh nodes along a real-time traffic's route to ensure a transmission with good QoS performance. The reservations are thus done for fix routes but repair mechanism are provided if a link fails and the route has to be changed. The DARE approach is implemented and tested in a simulation with ns-2.

Besides the simulation-based adaptation mechanisms, Guo et al. [8] implemented a mechanism called *Software-based Time Division Multiple Access* (STDMA) on top of the WLAN MAC layer in a testbed. The approach is designed to support WLAN-based VoIP appplications and it is claimed that a significant improvement of the maximum number of G.729-quality voice conversations in a WLAN is achieved. Typical scenarios with both best effort and real-time traffic are though not in the scope. This reduction to single use cases is, besides the MAC layer changes, the second difference to the approach presented here.

There are also propositions for QoE provisioning on higher layers. He et al. [9] introduce a middleware-based QoS control in 802.11 wireless networks. The idea is to implement a traffic prioritization inside the mesh nodes based on control theory. To realize this prioritization a "middleware design with cross-layer framework" is introduced and implemented in a Linux-based testbed. Above the middleware, the applications have the possibility to define requirements for single connections. Before a service is started, the application informs the middleware that certain QoS specifications are needed for the desired flow between two end points. The middleware's task is to choose an adequate service class on a dynamical base depending on the current performance of the service and the demanded requirements. By a control loop the current quality is measured and compared to the desired one. Depending on the current "quality error" dynamical packet scheduling is performed.

To distinguish our approach from [9] two things are mentioned. As the middleware approach is based on prioritization inside the mesh nodes, only problems caused by traffic passing through one of the nodes prioritizing multimedia streams can be handled. If the traffic problems occur due to collisions on the air interface caused by nodes that are not demanded to prioritize any real-time traffic among themselves, they will not recognize any problem and not control the disturber to solve the problem. There is no signaling mechanism between different nodes using the middleware software to locate a problem outside the real-time route. Depending on the focused field of application, there might be a second drawback of the approach presented in [9]: All services that need a certain QoS performance have to be announced first.

III. WLAN Mesh Networks and the MeshBed Setup

Wireless mesh networks are an interesting new approach to provide cheap, reliable, and flexible broadband wireless Internet access. As shown in Fig. 1, a WMN consists of a number of different devices connected over wireless links. A *Mesh Point* (MP) is a node which fully supports mesh relaying, meaning that it is capable of forming an association with its neighbors and forwarding traffic on behalf of other MPs. Besides these MPs, there are special *Mesh Access Points* (MAPs) which act as APs as well, connecting non-MP-capable devices to the WMN. A *Mesh Point Portal* (MPP) is another MP, bridging traffic between different WMNs or connecting the WMN to the Internet.

As todays technology and infrastructure developments have advanced, e.g. when looking at WMNs, the services used by the customers nowadays have as well. As for instance *Voice over IP* (VoIP) has become more and more popular, networks and mechanisms are necessary to ensure high quality for real-time applications. The performance of real-time applications in WMNs has been widely studied in terms of simulation, but only a few testbeds exist. We have investigated the possibility of real-time application support in a WLAN-based mesh network testbed, called "*MeshBed*", that has been developed by T-Systems in Darmstadt, Germany. Details about the *MeshBed* can be found in [10]. Fig. 1 shows a symbolic excerpt of this network. In case of the *MeshBed*, the single mesh routers are connected via WLAN on the 5 GHz frequency band. The gateway is connected to the core network providing Internet access via Ethernet. Access Points in the *MeshBed* are allowing notebooks, WLAN-based telephones, and other client devices to connect via Ethernet cable or WLAN on the 2.4 GHz frequency band.

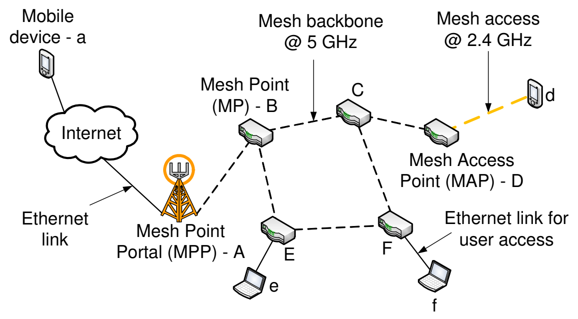


Fig. 1. *MeshBed* architecture

Currently, the *MeshBed* consists of 12 mesh routers and two mesh gateways, which are all deployed indoors. For investigations in more realistic scenarios, it is planned to extend the *MeshBed* with a 15 nodes outdoor mesh network. The mesh routers consist of embedded AMD Geode SC1100 Systems with 266MHz CPUs and 64 MB of RAM. The gateways consist of barebone desktop PCs with 3 GHz Intel Pentium 4 processors and 1 GB of RAM. All mesh nodes are equipped with Atheros Wireless Mini PCI WiFi Cards as well as Ethernet ports and use operating systems based on Linux together with madwifi [11], an open-source WiFi driver.

In the next section, the approach for QoS support in WMNs is presented.

IV. A Routing Layer based approach

A. Idea and General Structure

1) Idea of the Approach: The general idea of the approach is to perform the QoS support at the routing layer. MAC layer changes would be possible as well but they are not suited in this case. WLAN has already become a wide spread technology. Changing something in the MAC layer as currently standardized would not just mean an update to or recreation of all drivers for the WLAN devices but also implies possible hardware changes in those devices. This makes the deployment and usage of new MAC mechanisms very difficult.

Routing layer mechanisms to enhance QoS are a promising approach for WLAN-based mesh networks. The routing layer is easily exchangeable, as it is totally based on software. Independent of the operating system, the routing layer is logically situated on top of the network device driver and interacting with it via driver independent interfaces.

In the presented approach, maximal adaptability and flexibility is reached through a distributed solution. Every relay node is equipped with capacities to monitor, judge, and react on the current network situation.

The aim of the proposed mechanism is to keep track of the services currently present in the network. Approaching or already present problems shall be recognized as fast as possible. Solutions to those problems on different ways shall be provided to ensure a stable and high QoS level.

This aim basically needs two main tools to be realized, a *Traffic Observer* that analyzes the current network situation and a *Traffic Controller* that offers different possibilities to influence the actual situation to provide high QoS. Furthermore, an effective way to allow communication between those two components not only when present on one mesh node but also when distributed throughout the network is necessary. The following sections explain the different parts of the mechanism in more detail.

2) General Structure and Interoperability: Fig. 2 shows the general structure of the developed mechanisms. The core of the implementation is formed by the OLSR implementation of Andreas Tonnesen *OLSRd* [12]. Running on every node, this software enables the mesh routers to connect to each other and to form the *MeshBed*. The *Traffic Observer* is implemented as a kernel module. It is runnable independently of *OLSRd* and can be compiled and used on any linux machine with the correct kernel version. The *Traffic Controller* is implemented as a plugin to the *OLSRd* plugin interface. It includes a signaling unit making use of the *OLSRd* broadcast messages and allows thus communication between different *Traffic Controller*s. Located on one single node *Traffic Observer* and *Traffic Controller* are contacting each other via the Linux *netlink* sockets.

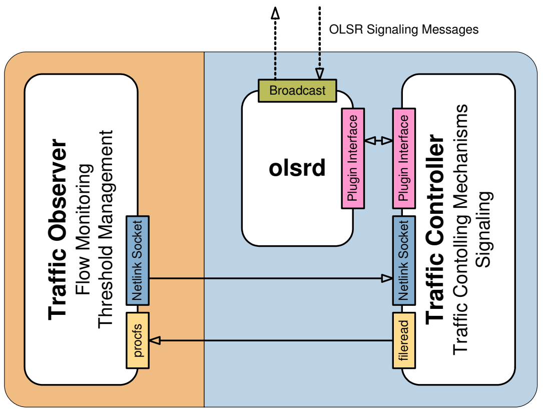


Fig. 2. General Structure

B. Traffic Observer

The key part of the presented approach is the component called *Traffic Observer*. Its tasks are two folded. On the one hand this module has to monitor the current situation in the mesh network by observing the current traffic flow, as well as other information that can be obtained from the network. On the other hand it has to judge whether the current network situation is acceptable or, if this is not the case, how to react on the occurring problems. To realize this, certain thresholds are needed. In the following sections each of these two tasks is presented in detail.

1) Flow Monitoring: As mentioned before, the most important task of the *Traffic Observer*, as the name says, is observing the network and the traffic inside it. Especially because *Traffic Observer* and *Traffic Controller* are normally situated in every relay node, there is much information of different kind that might be obtained and analyzed. In a raw classification one might separate this information into packet or traffic related information and non-packet or -traffic related information. Even though the latter one, including things like CPU usage or memory load at the monitoring node, might also be of big interest, the main focus lies on the former[1].

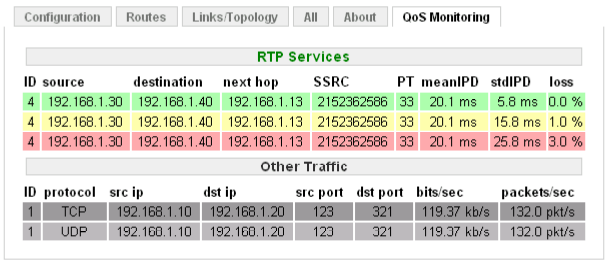
Traffic related information are all those information concerning the traffic of the network, i.e. the packets describing this traffic in the case of IP as in WLAN-based mesh networks. One of the main aims of the approach presented in this work is a distributed solution to the issue that is highly adaptable to different scenarios and network changes. This has a large impact on the possible choice of monitorable information. No information of neighbor nodes about their observations can be included in the measurements for two reasons. First, the standard packet structure of real-time services does not include any place to transport those information. Second, sending this information in separate packets with regular time intervals is impossible due to an insolvable trade off between too much signaling overhead and too imprecise information. OLSRv2 might solve this problem because it provides a more flexible signaling framework but it also increases the overhead and therefore it might not be a good approach especially in highly loaded networks with many active sessions.

All information the *Traffic Observer* can analyze about the currently active services is obtained by the observation of the packets passing by in the own node. Three different types of information can be obtained for a certain packet stream. First of all there is the explicit time independent information readable out of the packets content, as for instance source or destination address or protocol type. Next, there is the implicit time dependent information which is obtainable at the moment of the packet monitoring, e.g. the packet absolute arrival time or relative arrival time after the last packet of the same service. Finally, there is statistical information that is based on a series of packets rather than on a single one. This

[1]Normally it should be possible to choose the devices powerful enough in terms of memory and processor capacity so that those parameters do not become the bottleneck of a transmission. Limits of this estimation are shown in Section V.

information provides a long term analysis of the monitored services, for instance packet loss over the last n packets or the standard deviation of the packet inter arrival time. The measurement of the widely used one way delay metric is evidently not possible in this approach as information of more than one time stamp at other nodes in the network would be necessary. Though obtaining this information is impossible as explained before.

Fig. 3 shows a screen shot of the graphical information page displaying the information provided by the *Traffic Observer*. In the following section all displayed values are shortly described and assigned to the above classification. Furthermore, the equations to calculate the statistical information is given.



RTP Services

ID	source	destination	next hop	SSRC	PT	meanIPD	stdIPD	loss
4	192.168.1.30	192.168.1.40	192.168.1.13	2152362586	33	20.1 ms	5.8 ms	0.0 %
4	192.168.1.30	192.168.1.40	192.168.1.13	2152362586	33	20.1 ms	15.8 ms	1.0 %
4	192.168.1.30	192.168.1.40	192.168.1.13	2152362586	33	20.1 ms	25.8 ms	3.0 %

Other Traffic

ID	protocol	src ip	dst ip	src port	dst port	bits/sec	packets/sec
1	TCP	192.168.1.10	192.168.1.20	123	321	119.37 kb/s	132.0 pkt/s
1	UDP	192.168.1.10	192.168.1.20	123	321	119.37 kb/s	132.0 pkt/s

Fig. 3. A screenshot from the Browsers Monitoring Page

The information collected for Premium and RTP Services are as follows: source, destination, and next hop IP address of the packet can be obtained as explicit information, either out of the packet header, or in case of the next hop address out of the routing table by knowledge of the destination address. The payload type of the RTP service and its unique SSRC number are also explicitly readable from the packet header. The combination of SSRC and next hop address is used to assign a unique ID to each service. Packets with the same SSRC and next hop obtain the same ID and are collected together.

The values $mean_{IPD}$, std_{IPD}, and $loss$ are statistical information. To explain their calculation, the following definitions are given: For every packet p_i the following implicit and explicit information can be obtained:

ϕ_i: unique identification number of p_i,

t_i: absolute arrival time of p_i,

$\Delta t_i = \frac{t_i - t_{i-1}}{\phi_i - \phi_{i-1}}$: relative arrival time of p_i, and

l_i: total length of p_i in Bytes.

Furthermore, sets are held containing the obtained values for the last window size w packets $P = \{p_{last-w+1}, \ldots, p_{last}\}$ sorted by time of packet arrival:

$\Phi = \{\phi_{last-w+1}, \ldots, \phi_{last}\}$,

$T = \{t_{last-w+1}, \ldots, t_{last}\}$,

$\Delta T = \{\Delta t_{last-w+1}, \ldots, \Delta t_{last}\}$, and

$L = \{l_{last-w+1}, \ldots, l_{last}\}$.

Using these definitions, the statistical information can be obtained as follows:
The mean inter packet delay $mean_{IPD}$ is defined as

$$mean_{IPD} = mean[\Delta T] = \frac{\sum_{x \in \Delta T} x}{w}.$$

The standard deviation of the inter packet delay std_{IPD} is defined as

$$std_{IPD} = \sqrt{\frac{w}{w-1} \cdot \left(\frac{\sum_{x \in \Delta T} x^2}{w} - \left(\frac{\sum_{x \in \Delta T} x}{w} \right)^2 \right)}.$$

The packet loss $loss$ is defined as

$$loss = 1 - \frac{|\Phi|}{max[\Phi] - min[\Phi] + 1} = 1 - \frac{w}{max[\Phi] - min[\Phi] + 1}$$

The information collected for Other Traffic, i.e. non real-time traffic are as follows: The protocol type, source and destination addresses, and ports are explicit information of the packet header. The combination of source and destination addresses and ports are used to assign a packet to the correct monitored service. Bits/sec and pkts/sec are statistical information calculated as follows using the above definitions: The bandwidth in bits/sec bps is defined as

$$bps = \frac{\sum_{l \in L} l}{max[T] - min[T]}$$

The packet rate in pkts/sec is defined as

$$pktps = \frac{|L|}{max[T] - min[T]} = \frac{w}{max[T] - min[T]}$$

2) Threshold Management: The preceding section has offered a look inside the *Traffic Observer*'s monitoring facilities. It displayed which different types of information and parameters are measurable and how they are obtained. All information provided by the *Traffic Observer* is always available up to the most recent packet on demand via the Linux proc filesystem *procfs*.

Monitoring of the services alone is though not enough to do QoS monitoring and enhancement. There is also the need for a mechanism that judges the monitored information and reacts in the case of a possible quality decrease. To realize this task, a threshold management in the *Traffic Observer* is necessary. Following a common way of illustration, traffic light charts with colors green, yellow, and red depicting good, average, and bad quality are used.

Key parameters have to be compared to adequate thresholds to assign them with the correct color, i.e. quality level. The key parameters chosen in this work to judge QoS and a possible QoS degradation are the previously introduced std_{IPD} and $loss$.

In this work, the thresholds to do the QoS judgment on this parameters are configured service dependent. Each RTP payload type can be configured with four own values describing the $std_{IPD_{green-yellow}}$, $std_{IPD_{yellow-red}}$, $loss_{green-yellow}$, and $loss_{yellow-red}$ thresholds. One might imagine that thresholds could become less demanding in case of a larger number of services in the network or more claiming in an empty network. The thresholds defined in this work are though intentionally not adapting to different network situations. They are set to fixed values for every type of service.

As said before, the monitored values of the *Traffic Observer* are always available on demand via the *procfs*. More precisely, the explicit and implicit information for the w last packets are saved internally. At the moment of access to the *procfs*, the statistical information is calculated. The judged key parameters std_{IPD} and $loss$ belong to the statistical information as well. Nevertheless, they have to be compared to the thresholds regularly and not just on demand. std_{IPD} and $loss$ are thus calculated when $\lfloor \frac{w}{10} \rfloor$ new packets have arrived. For instance in case of $w = 100$ with the arrival of every 10th packet the std_{IPD} and $loss$ values are updated. Afterwards, the values are compared to the thresholds. If the thresholds are exceeded, a QoS alert is broadcast via the linux *netlink* socket. To avoid an alert flooding during the process of the reaction period, alerts are sent with an interval of 1 second.

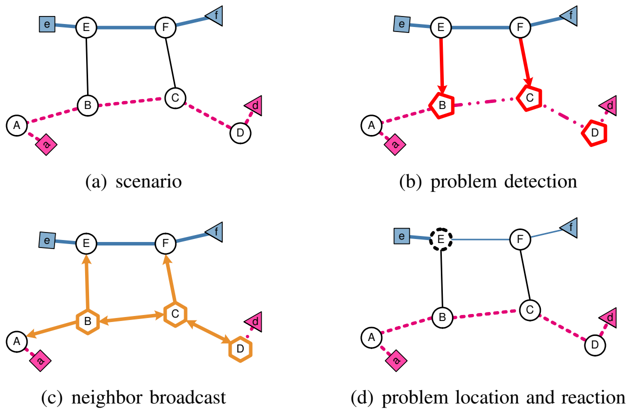
C. Traffic Controller

The second important unit of the mechanism is the so called *Traffic Controller*. So far, the possibilities of the *Traffic Observer* to detect a problem and its ways to give alerts have been presented. The remaining logical steps of the mechanism to solve quality problems are signaling the quality problems to other nodes in the *MeshBed* and to react on the disturbing influence to increase the quality. These tasks are realized by the *Traffic Controller* and are presented in this section.

1) Traffic controlling mechanisms: Quality degradation can occur for several reasons like packet loss, jitter, and long end-to-end delays. A common approach to decrease the packet loss and the jitter is packet prioritization using the type of service bit in the IP header. However, due to problems on the air interface caused by subsequent nodes when relaying traffic over multiple hops, a prioritization alone does not work in WMNs.

Considering the possibilities of automated and manual WLAN channel choice, it can be estimated that there are no external influences to the WMN on the air interface. All colliding packets are originating from one of the own mesh routers in the *MeshBed*. Under these circumstances a reaction to these collisions can be done by a reduction of the disturbing traffic's packet amount. By reducing the allowed bandwidth for non real-time traffic to a lower but still acceptable level, the frequency of possible disturbing packets is automatically decreased as well.

2) Steps of Controlling: Fig. 4 shows the steps of a *Traffic Controller* reaction in an example scenario inside the WMN environment displayed in Fig. 1. A constant bitrate real-time connection between a and d via A-B-C-D is disturbed by crossover high bandwidth traffic from e to f via E-F, see Fig. 4(a). The packets relayed from E to F and from F to f collide on the air interface with the packets relayed from B and C which results in a quality decrease of the real-time service, as illustrated in Fig. 4(b). The *Traffic Observers* at B, C, and D detect the quality problem and send an alert to their *Traffic Controllers*. At first the nodes try to find possible disturbances in their own queues. To avoid quality decrease caused by overloaded queues, all non real-time applications in the own node are checked first, if a certain bandwidth threshold is exceeded. If this is the case, the bandwidth of the non real-time applications is reduced. A bandwidth of 5 Mbps is supposed as sufficient for most purposes. In the used practical implementation, the *Traffic Controller* reduces the bandwidth to 5 Mbps in case of real-time problems. A dynamical stepwise adaptation of the bandwidth for non real-time traffic is an interesting topic to be researched and tested by simulation studies in future work.



(a) scenario (b) problem detection (c) neighbor broadcast (d) problem location and reaction

Fig. 4. Steps of Controlling

In the next step as neighbor nodes might cause crossover problems, as for instance E and F do in this scenario, signaling messages are sent to all one-hop neighbors via the *OLSRd* Hello Message system. This is shown in Fig. 4(c). All nodes receiving such a broadcast message of a disturbed node are as one-hop neighbors of the disturbed node possibly responsible for the disturbance. Therefore, they check and control the bandwidth of possible disturbing traffic the same way as the disturbed node did before. In the displayed scenario, E will activate the bandwidth control. F then recognizes that the bandwidth is already reduced to 5 Mbps and no further reaction is necessary. Fig. 4(d) shows the situation after the reaction of the mechanism. E is performing bandwidth control that leads to a slower but still working high bandwidth traffic from e to f. The performance of the real-time flows increases again and the QoS demands can be met.

V. Performance Measurements

To analyze the performance of the presented approach, the WMN environment and scenario as depicted in Fig. 1 and

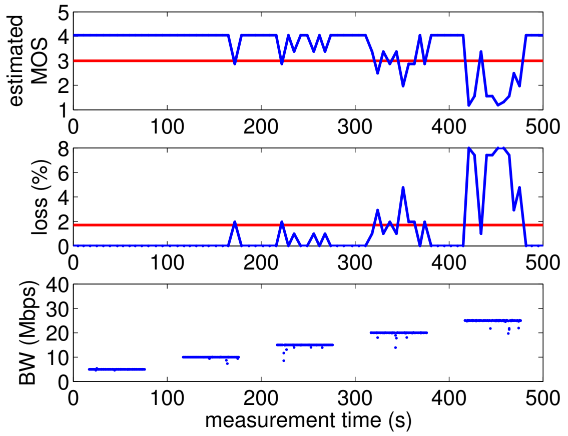


Fig. 5. Influences of Crossover Disturbers

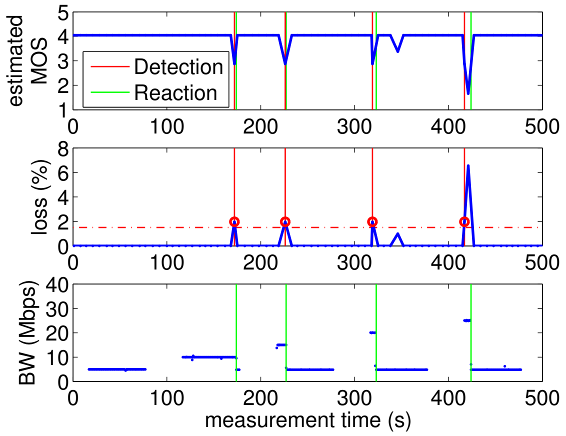


Fig. 6. Improvements by the *Traffic Controller*

Fig. 4 has been set up in a testbed. The constant bitrate real-time connection between a and d is realized by a VoIP connection with inter arrival time 20 ms and a packet size of 200 Bytes. This connection is disturbed by subsequent crossover high bandwidth connections from e to f via *E-F* with stepwise increasing bandwidths of 5, 10, 15, 20, 25 Mbps.

Fig. 5 and Fig. 6 successively show the results of measurements with deactivated and activated controlling mechanism. The x-axis shows the time of the measurement in seconds, the y-axes show the estimated *Mean Opinion Score* (MOS) [13] and the $loss$ in percent of the real-time traffic measured at *D* as well as the bandwidth in Mbps of the disturbing service measured at *F*.

The std_{IPD} has also been measured at *D*. However, the measurements have shown that even for the highest disturbing bandwidth of 25 Mbps, this parameter still stays in an acceptable level below 5 ms. Therefore, it is not displayed in the measurement results. The $loss$ value is on the other hand a lot more sensible to collisions on the air interface. As Fig. 5 shows, it is already sporadically increasing for a disturbing bandwidth of 10 Mbps.

A MOS value of less than 3, marked by a red line in Fig. 5, can be considered to imply bad quality. For $loss$ values bigger than 1.7 % the MOS goes below this threshold. This $loss$ value is thus also marked by a red line. Fig. 5 shows that the threshold is already exceeded for a disturber bandwidth of 10 Mbps. For disturber bandwidths of 20 Mbps and more, the quality is close to or below the accepted value during the whole period of disturbance. For the highest tested bandwidth of 25 Mbps, the service quality at *D* is totally unacceptable as the $loss$ value increases drastically.

Fig. 6 shows the same case as Fig. 5 but with activated mechanism at all nodes *A, B, C, D, E,* and *F*. Obviously, as a first perception, the phases with high $loss$, invoking low MOS, are a lot shorter than without the influences of the mechanism.

The vertical green and red lines in the curves show the time of the problem detection and the time of the controller reaction. The first exceeding values alerted at the time of the detection of a new problem are marked with a red circle in the $loss$ graph. The *Traffic Observer* threshold between yellow and red $loss$ values is set to 1.5 % and displayed in the graph by a dotted horizontal red line.

The bandwidth graph shows the reaction by reduction of the disturbers bandwidth to the configured value 5 Mbps. This obviously leads to a direct return to acceptable quality values in the $loss$ and MOS curves.

To quantify the performance of the mechanism, the key parameters, reaction time and signaling message load, have been analyzed. Depending on the number of neighbors a mesh router in the depicted scenario receives on average between 400 Byte, about 3 to 4 packets, and 2000 Byte, 15 to 20 packets, of *OLSRd* messages per second. As said before, the *Traffic Observer* does not send alerts more frequently than with an interval of 1 second to avoid an alert flooding. An alert is furthermore broadcast by an *OLSRd* message of a size fitting in one single *OLSRd* packet. This one additional packet per second does not show any increase of the average *OLSRd* signaling bandwidth. Even the highest measured *OLSRd* signaling bandwidth of 2000 kbps is ignorable even in a highly loaded network. The signaling load issue is thus no problem of the presented mechanism.

The second important metric to quantify the mechanism's performance is the reacting time. As upcoming quality loss is recognized latest within the first w disturbed packets, i.e. in the default case with $w = 100$ and constant bitrate $20\,ms$ in the first two 2 seconds, the delay between the occurrence of a quality decrease and the recognition can be disregarded. Then again an activation of the *Traffic Controller*, e.g. reducing the disturbers bandwidth, is supposed to solve the problem in maximally w packets as well, what can be confirmed by a look at Fig. 6. The time between the activation of the *Traffic Controller* and the return of an acceptable quality level is thus also negligible. The scope lies on the delay between the detection and the reaction. Fig. 6 shows that this delay depends on the bandwidth of the disturber. For the bandwidths of 5, 10, 15 Mbps the delay is between 1 and 3 seconds. Such a delay results only in a short QoS loss which is still acceptable for a user.

For the test cases with higher bandwidths of 20 Mbps and 25 Mbps, which are though not expected to occur in real mesh networks, the delays increase significantly up to 7 seconds. An analysis of the single controlling steps has shown that the high delays in the measurement setup are mainly caused inside the *Traffic Controller* while activating the traffic reduction. The delays are due to high CPU use of the used mesh nodes. In this case the prerequisite of Section IV that the nodes can be chosen fast enough to not be the bottleneck of a transmission is not met anymore with the used equipment. The effects are though expected to disappear when more powerful machines or a hardware based realization are used. Investigating such a realization might be a promising topic for future work.

Testing the efficiency of the mechanism in other network situations and a design of experiments for different network factors like number of nodes, number of connections, network load, and so on is difficult in a practical implementation and implies simulation. Implementing the approach in a simulation environment to obtain more information about efficiency and general usability might be thus interesting for future work.

VI. Conclusion

In this paper, we presented a measurement-based approach to support real-time applications in wireless mesh networks. In contrast to other publications in this area, the developed algorithm was not just tested in a simulation environment, but implemented in a real WLAN-based mesh network. It was shown that the developed concept works well in a real implementation.
The approach is based on two main entities, a *Traffic Observer* and a *Traffic Controller*. Whenever the *Traffic Observer* detects a problem in the mesh network, for example a high rate best effort flow blocks a real-time application, the *Traffic Controller* forces this low priority flow to reduce its bandwidth. The results have shown that the mechanisms reacts in less than three seconds which is completely sufficient for real-time traffic over WMNs. The next step is to improve the performance of the developed system using several different scenarios and to extend our mechanism by an efficient admission control scheme for real-time traffic flows. Also the simulative investigation of adaptation mechanisms for non real-time traffic is one topic for future work.

References

[1] IEEE 802.11s/D1.0. Draft Amendment to Standard for Information Technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - ESS Mesh Networking, March 2007. IEEE 802.11s/D1.

[2] IEEE 802.15 Standard Group Web Site. Available from: http://www.ieee802.org/15/.

[3] IEEE 802.16j Mobile Multihop Relay Project Authorization Request (PAR), Official IEEE 802.16j Website: http://standards.ieee.org/board/nes/projects/802-16j.pdf, March 2006.

[4] Rastin Pries, Stefan Menth, Dirk Staehle, Michael Menth, and Phuoc Tran-Gia. Dynamic Contention Window Adaptation (DCWA) in IEEE 802.11e Wireless Local Area Networks. In *The Second International Conference on Communications and Electronics, HUT-ICCE*, Hoi An, Vietnam, June 2008.

[5] Hao Zhu, Guohong Cao, Aylin Yener, and Allen D. Mathias. EDCF-DM: A Novel Enhanced Distributed Coordination Function for Wireless Ad Hoc Networks. In *IEEE ICC 2004*, pages 3886–3890, Paris, France, June 2004.

[6] Lamia Romdhani, Qiang Ni, and Thierry Turletti. AEDCF: Enhanced Service Differentiation for IEEE 802.11 Wireless Ad-Hoc Networks. In *IEEE WCNC*, 2003.

[7] E. Carlson, C. Prehofer, C. Bettstetter, H. Karl, and A. Wolisz. A Distributed End-to-End Reservation Protocol for IEEE 802.11-Based Wireless Mesh Networks. *IEEE Journal on Selected Areas in Communications*, 24(11):2018–2027, November 2006.

[8] Fanglu Guo and Tzicker Chiueh. Comparison of QoS Guarantee Techniques for VoIP over IEEE802.11 Wireless LAN. In *Multimedia Computing and Networking 2008*, San Jose, CA, USA, January 2008.

[9] Wenbo He, Hoang Nguyen, and Klara Nahrstedt. Experimental Validation of Middleware-based QoS Control in 802.11 Wireless Networks. In *BROADNETS '06: Proceedings of the Third International Conference on Broadband Networks*, pages 1–9, San Jose, CA, USA, 2006.

[10] Nico Bayer, David Hock, Matthias Siebert, Andreas Roos, Bangnan Xu, and Veselin Rakocevic. VoIP performance in MeshBed - a Wireless Mesh Networks Testbed. In *Proc. IEEE 67th Vehicular Technology Conference (VTC '08-Spring)*, Marina Bay, Singapore, May 2008.

[11] http://www.madwifi.org.

[12] Andreas Tonnesen. Implementing and extending the Optimized Link State Routing Protocol. Master Thesis at UniK, 2004.

[13] ITU-T. ITU-T Recommendation P.800, Methods for subjective determination of transmission quality. ITU-T Recommendation, 1996.