European Collaborative Innovation Centres for Broadband Media Service

QoS-Aware Channel Scheduling for Multi Radio/Multi-Channel Wireless Mesh Networks

Case Study: Performance Optimizations for Voice over IP in Wireless Mesh Networks

Investing in the future by working together for a sustainable and competitive region

EUROPEAN UNION
European Regional Development Fund

The Interreg IVB North Sea Region Programme

1 Abstract

In non-static multi-radio/multi-channel wireless mesh networks architectures such as Net-X, mesh nodes need to switch channels in order to communicate with different neighbors. Present channel schedulers do not consider the requirements of real time traffic such as voice over IP. Thus the resulting quality is low. We propose a novel channel scheduler for the Net-X platform that takes into account packet priorities. We evaluate the algorithm on the KAUMesh testbed. Our algorithm outperforms the standard round-robin scheduler both in terms of average delay and jitter.

2 Introduction

Wireless mesh networks (WMNs) are considered to be promising technology for cost-efficient proliferation of internet access in both sparse rural areas as well as in densely populated urban areas. In a WMN mesh routers relay traffic on behalf of clients or other routers and by this form a wireless backbone.

While in the first generation of mesh networks routers only used one radio and one channel to forward traffic, in the second generation multiple radios and channels are used simultaneously. This can increase the network capacity drastically, but also adds extra complexity.

For example, if a node uses more channels than it has cards available, then multi-channel operation imposes the need for channel switching. According to [8] the channel switching time on current hardware is between $200\mu s$ and $20ms$, which causes a very high overhead for per-packet switched channels. The channel-switching-time is composed of several phases: sending buffered frames in the hardware queue of the NIC, stopping interrupt service routines of the driver, tuning to the new frequency, re-starting the interrupt service routines and sensing the medium.

While some of those phases can be shortened significantly by clever hard- and software-design, others are more challenging to optimize. For example, *per-channel* hardware buffers can store frames accross channel switches and thus flushing the buffer prior to the switch is not needed. Also an optimized design of the RF-filter mask tuning improves the performance.

In the standard IEEE 802.11 DCF protocol collision avoidance relies to a large part on correctly detecting the medium status by the means of channel clear assessment and/or RTS/CTS. Those functions only work well if a node can sense the medium long enough to hear frame-preambles or RTS/CTS frames [13]. Thus, a node should not immediately transmit data after a channel switch. We believe that this is a fundamental shortcoming of the IEEE 802.11 MAC protocol in channel-switched systems, which needs more investigation. As a result, even good hard- and software design do not allow per-packet channel switching with an acceptable overhead when IEEE 802.11 DCF is applied.

Therefore, multi-radio multi-channel mesh network platforms such as Net-X [4] use per-channel queues, which are serviced for a longer time span. A scheduler then has to decide when and for how long a channel is serviced.

For delay sensitive traffic such as Voice over IP, the right scheduling strategy is crucial for providing good end-user quality. Important information such as traffic priority should be considered as input parameter to the scheduling approach. In this paper we address the issue of a QoS-aware channel scheduling based on traffic priorities. The key-contributions are:

- A novel QoS-aware scheduling algorithm.
- An analytical evaluation of the algorithm.
- An implementation of the algorithm in the Net-X platform and a performance evaluation.

The remainder of this paper is organized as follows: In Section 3 we introduce the multi-radio/multi-channel framework Net-X and discuss related work. The design and implementation of our proposed scheduler is presented in section 4. Section 5 compares the performance of our scheduler to a base-line round-robin scheduler. Finally, Section 6 presents our conclusions and suggests future improvements.

3 Background and Related Work

3.1 Multi-channel Mesh Networks

Multi-channel protocols and architectures are designed to exploit the available channels to enhance the overall throughput. According to [5], the classification of channel assignment protocols can be based on how frequently the channel assignments are performed, therefore the protocols and architectures for multi-radio/multi-channel networks can be classified as dynamic, semi-dynamic, static and hybrid. Net-X [4] is an hybrid approach, as it applies a semi-dynamic assignment to the its fixed interface (used primarily for receiving data from neighbors) and a dynamic assignment to the switchable interfaces (used to transmit data to its neighbors). The semi-dynamic reassignments to the fixed interfaces is based on the current number of nodes using the same fixed channel. Therefore, if a node notices that the number of nodes using the same fixed channel as itself is large, it can reassign its interface to a less used channel and inform its neighbors.

The channel used by the switchable interface may be changed at any time, without having to inform the neighbors. Thus, the switchable interface can be used to transmit to neighbors whose fixed interfaces may potentially be on different channels. Net-X protocol stack includes a channel management module that determines which channel to assign to each fixed interface, and when each switchable interface may switch its channel. By judicious use of channels, it is possible to efficiently utilize a large number of channels in the mesh, even though each node is equipped with only two interfaces. Since the interface channel-switching may incur a non-negligible delay, a queuing algorithm to buffer packets is deployed in Net-X, as well as a round robin scheduling policy to transmit buffered packets in order to reduce frequent switching.

As shown in [12], unnecessary delays are created in Net-X due to the hello packets being transmitted consecutively on all the channels. For example, an interface that services four channels must switch to three other channels and spend at least a minimal amount of time on each channel before returning to the loaded channel. In order to avoid such unnecessary delays [12] proposes to replace the round robin scheduler by the ”delay sensitive” channel scheduler. The objective of the proposed scheduler is to service more often on the channels that have a higher average queue length. This is done in [12] by staggering the creation of the hello packets due to the overall benefit of servicing queues that have a longer average length. It is important to note that this solution does not make differentiation among service priorities, therefore if two channels are loaded (eg. one with VoIP traffic and another with TCP traffic), the scheduler will serve them equally.

3.2 IEEE 802.11e EDCA

IEEE 802.11e is a standard for QoS over IEEE 802.11 based networks (see for example [9]). Among other extensions to the original IEEE 802.11 MAC layer, it includes the Enhanced Distributed Channel Access (EDCA). EDCA allows the priorization of frames inside a node and among nodes. Inside a node, for each service class a queue is created, which holds packets of its service class. Queues are served by a virtual contention resolution mechanism similar to DCF. Furthermore, among nodes the using different inter-frame times can prioritize frames.

Initially, IEEE 802.11e was not designed for multi-channel/multi-radio networks. For example, IEEE 802.11e does not consider channel switching. Also, inside a node, each network card runs its own instance of the MAC protocol, not exploiting information possibly available from other network cards. However, IEEE 802.11e can be used as a complimentary design element in multi-channel networks. If, for example, several nodes compete for the same channel, IEEE 802.11e can still be useful to prioritize the medium access. IEEE 802.11e and the proposed scheduler are orthogonal approaches. In this paper we concentrate on the scheduler design and evaluation, but we consider to combine both techniques in a future work.

4 QoS Aware Channel Scheduler

4.1 Design Goals and Motivation

Existing schedulers do not take into account the priority of packets. Thus it can happen, that packets with smaller delay budget are delayed unnecessarily long, because a channel with delay-insensitive packets is served before. For delay sensitive traffic such as VoIP this will reduce the perceived quality. The goals of our scheduler are thus to minimize the delay and waiting time for delay sensitive traffic while at the same time providing reasonable throughput for delay-insensitive traffic at reasonable switching cost.

4.2 Scheduling Algorithm

Our scheduling algorithm selects the next channel based on the priority of the current channel and the priority of all other channels, which have packets to send. We assign the priority to a channel according to the priority of packets which are queued to be sent for this channel. If packets with different priorities are queued, the highest priority among all packets is used. The currently used channel might not have packets queued. In this case the priorities of packets since the last channel switch are taken into account. A packet's priority is determined by its DiffServ Code Point [10]. While DiffServ allows the definition of multiple traffic classes, for simplicity we only consider a low and high priority traffic class. The sender or ingress-router marks delay-insensitive TCP-traffic with low priority, realtime traffic such as VoIP with high priority. The concept could be extended to multiple priorities, which are even dynamically assigned (for example based on available delay budget).

The scheduling algorithm is sketched in code listing 1 and is composed of two parts. The first part (lines 1-8) determines which priorities the current (`getChannelPrio(Current Channel)`) and the next channel have. For every priority, the algorithm maintains a counter $served[prio]$ that represents the number of times that the priority was served. If the counter exceeds the per-priority configurable threshold $T_{u,prio}$ a new lower target priority ($target\,prio$) is selected and the current counter is reset to zero. If there are no channels of the target priority available, the algorithm looks for the next lower priority (`getTargetPrio(prio)`). Upon reaching the lowest possible priority, the algorithm starts over with the highest priority. In the second part (lines 9-12), among all channels of the new priority (`getChannelsByPrio(targetprio)`), the channel which has not been served longest is selected (using `getOldestChannel(CandidateSet)`).

When a channel is selected it is scheduled for a time of T_{min}. Furthermore if after T_{min} there are still packets available to send, the channel gets an extra time of $T_{defer,prio}$. After this time the channel scheduler is called again to select a new channel. In the rest of this paper we assume backlogged traffic, i.e. a channel is always scheduled for a service time $S = T_{min} + T_{defer,prio}$. And T_{min} and $T_{defer,prio}$ are configurable. Switching from one channel to another requires T_s delay.

Input: Current Channel
Output: Next Channel
1 prio ← `getChannelPrio(`*Current Channel*`)`;
2 targetprio ← `getChannelPrio(`*Current Channel*`)`;
3 **if** *served[prio]* $> T_{u,prio}$ **then**
4 | targetprio ← `getTargetPrio(`*prio*`)`;
5 **end**
6 **if** *prio* $!=$ *targetprio* **then**
7 | $T_{u,prio} \leftarrow 0$;
8 **end**
9 CandidateSet ← `getChannelsByPrio(`*targetprio*`)`;
10 Next Channel ← `getOldestChannel(`*CandidateSet*`)`;
11 $T_{u,target\,prio}++$;
12 **return** Next Channel;

Algorithm 1: QoS Aware Channel Scheduling Algorithm

4.3 Analysis

In this section we analyze the scheduling performance when two priorities are used. For example, consider a node has to schedule two low priority channels (L_1 and L_2) and two high priority channels (H_1 and H_2). The channel hopping pattern with our algorithm will be $H_1H_2L_1H_1H_2L_2$ and so forth ($T_{u,L} = 1$ and $T_{u,H} = 2$). In contrast the round robin scheduler produces a pattern like $H_1H_2L_1L_2H_1H_2L_1L_2$.

With regard to those two hopping patterns one can see that the waiting time for high priority traffic and the throughput for background traffic depends on the switch patterns and the values of T_{min} and $T_{defer,prio}$. In the following sub-sections we will first derive equations for the channel waiting time and throughput. Then we will apply those models to an example traffic pattern.

The analysis is based on the following assumptions:

- backlogged traffic
- two different packet priorities
- every channel only serves packets of one priority
- arbitrary number of high and low priority channels
- a fixed data rate

4.3.1 Waiting Time

The waiting time is the maximum time that a channel needs to wait until it will be rescheduled again. The waiting time has a direct influence on the packet delay. For a simple case with m high priority (H) and n low priority (L) channels the waiting time $WT(H)$ for a high priority channel is given in equation 1. It is the sum of the service-times all low and high priority channels that need to be served before the initial channel is serviced again. Please note that the service times $S(H)$ and $S(L)$ can be different for the QoS-aware scheduler, since it allows to have per-priority $T_{defer,prio}$. However for the round-robin scheduler $S(L)$ equals $S(H)$.

$$WT_{QOS}(H) = (T_{u,H} - 1) * S(H) + T_{u,L} * S(L) + (T_{u,L} + T_{u,H}) * T_s \quad (1)$$

Equation 2 calculates the waiting time for a round-robin scheduler.

$$WT_{RR}(H) = (m-1) * S(H) + n * S(L) + (m+n) * T_s \quad (2)$$

4.3.2 Throughput

Apart from the waiting time for the high priority traffic, the throughput of the background traffic is also important. We assume that the throughput for one channel is proportional to the amount of time scheduled for this channel. We are aware that the throughput is dependent on other factors such as the number of collisions and the transport protocol as well. Yet we believe that channel time is a sufficiently good measure of achievable

throughput in our context. Through the use of channel diversity the number of collisions and neighbors competing for the same channel is reduced.

The cycle length of a channel switch pattern is the number of priority changes after which the same pattern repeats again. For the QoS-aware scheduler the cycle length is

$$CY_{QOS} = lcm(\frac{lcm(m,T_{u,H})}{T_{u,H}}, \frac{lcm(n,T_{u,L})}{T_{u,L}}) \tag{3}$$

Here, $lcm(m,n)$ denotes the least common multiple of m and n. It calculates when a pattern is repeated. If $T_{u,H}$ or $T_{u,L}$ are 0, the cycle time is m or n respectivily. Using this equation we express the ratio of the service time within a cycle of a specific class (H or L) and the total cycle length as

$$CT_{QOS}(L) = \frac{CY_{QOS} * T_{u,L} * S(L)}{CY_{QOS} * (T_{u,L} * (S(L) + T_s) + T_{u,H} * (S(H) + T_s))} \tag{4}$$

and

$$CT_{QOS}(H) = \frac{CY_{QOS} * T_{u,H} * S(H)}{CY_{QOS} * (T_{u,L} * (S(L) + T_s) + T_{u,H} * (S(H) + T_s))} \tag{5}$$

For the round-robin scheduler the amount is given by

$$CT_{RR}(L) = \frac{n * S(L)}{m * (S(H) + T_s) + n * (S(L) + T_s)} \tag{6}$$

and

$$CT_{RR}(H) = \frac{m * S(H)}{m * (S(H) + T_s) + n * (S(L) + T_s)} \tag{7}$$

4.3.3 Example

Using the equations from the previous section we compare the throughput and the waiting time of our QoS-aware scheduler and the round-robin scheduler. The example makes use of some system and traffic pattern parameters which are used in the real system evaluation described later on in Section 5. The parameters and their values are listed in Table 1.

T_s represents the time, which the network card needs to tune to the new channel. It depends on the card chipset and the driver and is around 4 ms for our NIC and driver. As discussed in section 2, this value is dependent on many factors and cannot be arbitrarily small. The minimal scheduling time in a channel, T_{min}, is also limited by the implementation factors (e.g. interrupts, timer accuracy and overhead). The standard Linux timers have an accuracy of about 2 ms [6]. Considering this and the 4 ms overhead involved in every channel switch, we choose a T_{min} of 15 ms as a compromise between switching overhead and delay. m and n indicate the number of low- and high priority channels and are a direct result of the traffic pattern.

In contrast to the previous parameters, that are *implementation-dependent*, $T_{u,L}$, $T_{u,H}$, $T_{defer,L}$ and $T_{defer,H}$ are configurable parameters. We chose the values such that high priority traffic is served more often ($T_{u,H} > T_{u,L}$) and low priority traffic is serviced longer ($T_{defer,L} = 10\ ms$) each time. Those settings are specific for the given traffic pattern. By using these parameters, Table 2 shows that the waiting time WT(H) for QoS-aware scheduler is around 29 ms lower than the one for the round-robin scheduler.

T_s	4 ms
T_{min}	15 ms
$T_{u,L}$	1
$T_{u,H}$	2
$T_{defer,L}$	10 ms
$T_{defer,H}$	10 ms for RR, 0 for QoS
m	2
n	2

Table 1: Algorithm Parameters

Value	QoS Scheduler	Round Robin Scheduler
CT(H)	44.8%	43.1%
CT(L)	37.3%	43.1%
Switching Overhead	17.9%	13.8%
WT(H)	52 ms	81 ms

Table 2: Comparison QoS and Round-Robin Scheduler

However, the QoS-aware scheduler gives the low priority traffic a lower time share. Thus the throughput of the background traffic should be lower for the QoS-aware scheduler.

There is a trade-off between achieving small waiting time for high priority traffic and guaranteeing low priority traffic throughput. This trade-off can be controlled by $T_{defer,L}$. For the given configuration, the waiting time for high priority traffic equals $T_{defer,L} + 45$. Figure 1 depicts this trade-off and compares it to the default round-robin scheduler. It is important to note that we just vary the configuration parameters of the QoS scheduler (given by the solid line), and compare it to the waiting time for high priority traffic and low priority channel time share results of the round-robin scheduler presented in Table 2 ($WT(H)$ and $CT(L)$, given by the intersection of the dotted lines).

The graph can be divided into three regions. In the first region (waiting time $< 59ms$), the background traffic has lower time share and the high priority traffic lower delay as compared to the round-robin scheduler. In the second region (waiting time $> 59ms$ and $< 81ms$) the background traffic gets more time share and the high priority traffic has a lower waiting time. In the third region the QoS-aware scheduler gives the background traffic more channel time on cost of a higher waiting time for high priority traffic. To summarize, in the second region the QoS-aware scheduler is always better than the round-robin scheduler. In the first and third region, the operator can sacrifice either delay (of high priority traffic) or throughput (of low priority traffics).

For arbitrary traffic patterns, equations (1)-(7) allow to quantify the trade-off between throughput and waiting time similarly to Figure 1 and how it behaves by varying $T_{defer,L}$. However not considered in this work, a natural extension to our algorithm is the selection of the parameters based on a higher-level objective, such as the ratio between gain of latency (smaller channel waiting time) versus loss of throughput compared to the standard RR-scheduler.

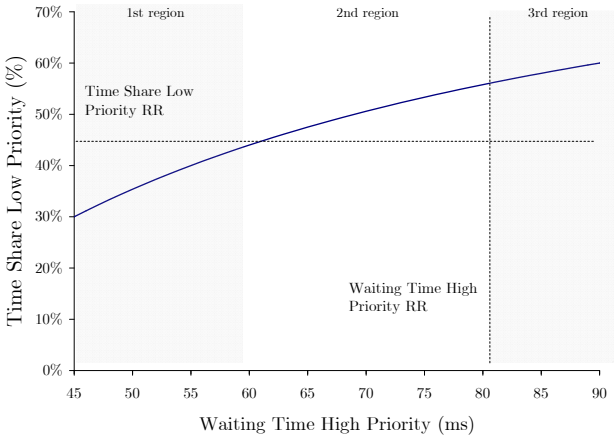


Figure 1: Trade-off Between Delay and Throughput

4.4 Implementation

We implemented our algorithm by extending the bonding driver from the Net-X platform. A hybrid multi-channel protocol is used in order to couple the destination address and the channel used to reach this destination [7]. Figure 2 shows an architectural view of the bonding driver. When packets arrive from the user space or the routing layer they are assigned a priority based on the Diff-Serv field in the IP-header. There is one queue for every channel. If the card is already tuned to the correct channel, packets are directly forwarded to the network card driver. Otherwise packets are placed in their respective queues. The scheduling algorithm is called by a timer when the service time for a channel ends and selects the next channel queue to service based on algorithm 1 or a round-robin scheme.

5 Performance Evaluation

5.1 Evaluation Environment

We evaluated our algorithm on the KAUMesh test bed[2], which is an indoor wireless mesh network with 20 nodes deployed at House 21 at Karlstad University. The nodes are based on the Cambria GW2358-4 platform, Linux 2.6.22, MadWifi 0.9.4 and Net-X. Every node is equipped with three Atheros-based wireless network interface cards. We aim to minimize wireless link variability factors, therefore the PHY-rate and the routes are fixed throughout the evaluation. Two interfaces are used for the mesh-backbone operating in IEEE 802.11a mode and and a PHY-rate of 6 Mbps. The third interface is used for client access in 802.11b/g mode. By using the IEEE 802.11a channels on the

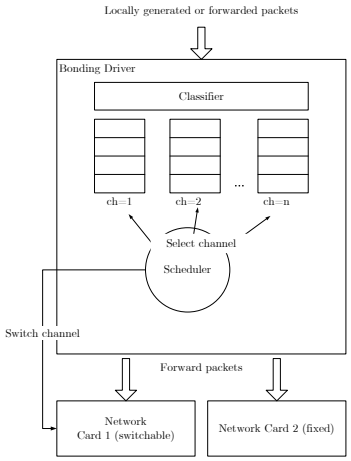


Figure 2: Bonding Module

backbone, which operate in the 5 GHz-band, we avoid interfere with the campus WLAN network operated on the 2.4 GHz-band.

Figure 3 depicts the network topology used for the evaluation. It consists of five nodes each mounted in the ceiling of lab rooms or corridors in House 21. The topology is simple and controlled to avoid problems such as route breaks, re-routing and bad links, which will typically occur as the network grows. Node 7 runs an NTP-server. All other nodes synchronize their hardware clocks against the NTP-server every three seconds. By this we achieve a clock skew smaller than 1 ms. Please note that the clock synchronization is not needed for the scheduling, but only for the timestamps used in the delay measurements.

In our experiments the VoIP traffic is generated by mgen tool [3] using 200 UDP datagrams per second (CBR) of 168 bytes between nodes 7 and 11 in both directions. This traffic pattern simulates four concurrent VoIP calls with the G.711 codec. At the same time, the TCP traffic is generated with the iperf tool [1] between nodes 10 and 21, and nodes 10 and 23, as shown in Figure 3. Therefore, for the given scenario, the number of high and low priority channels is equal to 2 ($m = 2$ and $n = 2$, respectively). Each experiment run lasted for 55 seconds.

5.2 Results

Latency, jitter and packet loss are key characteristics to be considered while investigating the performance of voice traffic over multi-radio multi-channel wireless mesh networks networks. In this section we present the behavior of these metrics while ana-

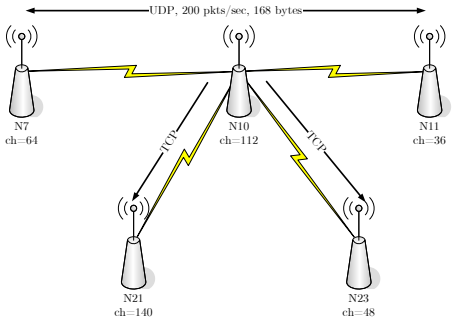
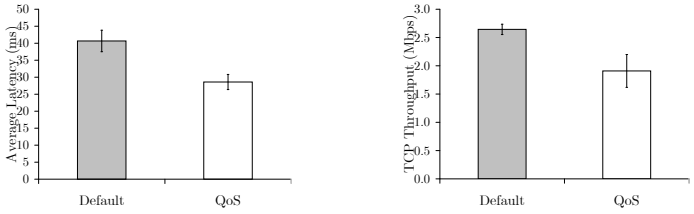


Figure 3: Network Topology



(a) Latency (b) TCP Throughput

Figure 4: Average Latency and TCP Throughput

lyzing the scheduling algorithm. The results represent the average over five repetitions, while the error bars represent the respective standard deviations.

Understanding latency characteristics is crucial for delay sensitive applications such as VoIP. According to [11] to obtain a good voice quality the delay imposed by the network should stay below 150msec, since VoIP user should not tolerate excessive delays in conversation. The results in Figure 4(a) show the average latency experienced by the two scheduling strategies studied: round robin scheduling (default) and the QoS-aware channel scheduling (QoS). We observe that the QoS-aware scheduler decreases the average latency of the VoIP packets if compared to the round robin scheduler. This is because the proposed scheduler guarantees that high priority channels carrying VoIP traffic are preferred in relation to the lower priority channels carrying TCP flows, which consequently reduces the average TCP throughput as shown in Figure 4(b).

Similarly, excessive jitter makes the service unusable by negatively impacting service quality. Since jitter is categorized as the change in latency from packet to packet, we plot in Figure 5 the latency experienced by the VoIP packets along the measurements. The X-axis represents the VoIP packets' IDs of one UDP flow between nodes 7 and 11.[1] and the Y-axis represents the packet delay in seconds. The graphs show the behavior of packet's delay (Y-axis) along the packets generated (X-axis). The spikes represent the most delayed packets that are waiting on the node's channel queue to

[1] Due to space constrains, we plot a range of 300 packet IDs.

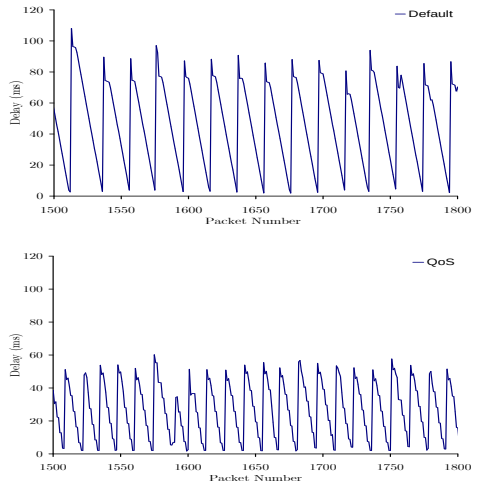


Figure 5: Voice Packet Delay versus Voice Packet IDs

be transmitted. In contrast, the least delayed packets are the packets not queued, as the node currently accessing the right channel. We see that by using the round robin scheme, some VoIP packets can achieve delay of 100 ms or more, in contrast with 60 ms while using the QoS-aware scheme. The network jitter in both schemes is because of the delay that certain (most delayed) packets experience in the nodes' queues while waiting for the next channel transmission opportunity. As expected, our scheme reduces VoIP latency, and therefore network jitter, as the VoIP's channels have higher priority during the scheduling decision than the TCP's channels.

If compared to the waiting time presented in Table 2, the measured waiting time values of 60 and 100 ms obtained are within a small error in accordance with the theoretical analysis from section 4.3, since in the theoretical analysis the time for transmitting the packet over the air and to processing it at the receiver are not considered.

To better explain the previous results, we plot in Figure 6 the channel hopping pattern for the two schedulers studied. The channel hopping patterns are obtained through the analysis of the channel switching dynamics in node 10's switchable interface. The X-axis illustrates the measurement time in milliseconds and the Y-axis illustrates the 802.11a channels used by the switchable interface. By considering the topology in Figure 3, we can point out that the channels 36 and 64 are the high priority channels (VoIP packets) and the channels 48 and 140 are the low priority channels (TCP packets) used. According to the results, by using the round robin scheme, the average waiting time for the high ($WT_{RR}(36)$ and $WT_{RR}(64)$) and the low ($WT_{RR}(48)$ and $WT_{RR}(140)$) priority channels were 79, 81, 70 and 68 ms, respectively. However, by using the QoS-aware channel scheme, the average waiting times for the high ($WT_{QoS}(36)$ and $WT_{QoS}(64)$) and the low ($WT_{QoS}(48)$ and $WT_{QoS}(140)$) priority channels were 56, 47, 128 and 101 ms, respectively. In Figure 6 we can also observe the influence of defer-time for the high ($T_{defer,H}$) and low ($T_{defer,L}$) priority channels presented in Table 1. For the round robin

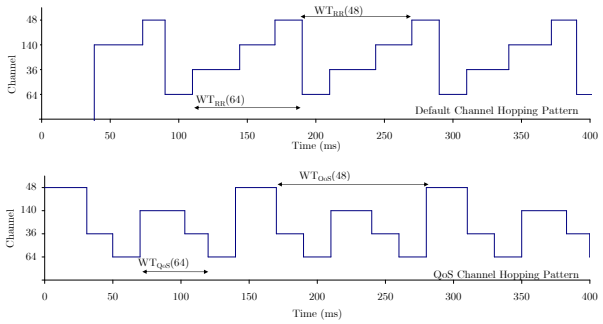


Figure 6: Channel Hopping Pattern for the Default and QoS Scheduler

scheduler, the high and low priority channels make use of the defer-time and therefore the channel service time can be extend by 10 ms. For the QoS scheduler, the $T_{defer,H}$ is not used and therefore just the low priority channels experience a higher channel service time.

Applications and end-user devices are designed to tolerate a certain amount of jitter. This is achieved through the so called jitter buffer, by buffering the data flow and designing processing algorithms to compensate for small changes in latency occurring from packet to packet. Depending on the application, the tolerable amount of jitter will vary. For example, a VoIP service should have a jitter buffer of approximately 80 ms or less [11]. By achieving a lower waiting time for high priority traffic using the QoS-aware scheduling, we also guarantee the reduction of the jitter buffer size required by the nodes. In order to verify that, we plot in Figure 7 the histogram and the cumulative distribution function (CDF) of the measured delay of VoIP packets for one test run and both schedulers. The X-axis represents the packet delays and the Y-axis represents the normalized number of packets mapped inside each interval. The Y-axis in the right hand-side presents the CDF of the packet delays. From the CDF we note that for the round robin scheduler more than 40% of packets experience a delay larger than 50 ms. In contrast, for the QoS scheduler only 8% of the packets experience such delay. The histogram also shows that the majority of the packets experience a maximum delay of 70 ms and 100 ms for the QoS and default scheduler respectively.

Packet loss can occur due to several factors, such as medium congestion and high traffic load. The use of a jitter buffer also augments the amount of packet loss since packets with a delay difference greater than the selected jitter buffer size are also considered lost. Packet loss, then, can significantly reduce quality of service. We show in Figure 8 the relationship between the loss ratio of voice packets versus jitter buffer size for both schedulers. The packet loss ratio consists of packet loss introduced by the network and the jitter buffer. The packet loss introduced by the network in our measurements is 2.8$\pm$1.6% and 1.3$\pm$0.5% for the default and the QoS-aware scheduler respectively. We assume a simple, static jitter buffer that drops packets with an packet inter-arrival time difference greater than a certain threshold (shown on the X-axis). It is clear that the introduction of a greater jitter buffer size (e.g. 100 ms) may decrease

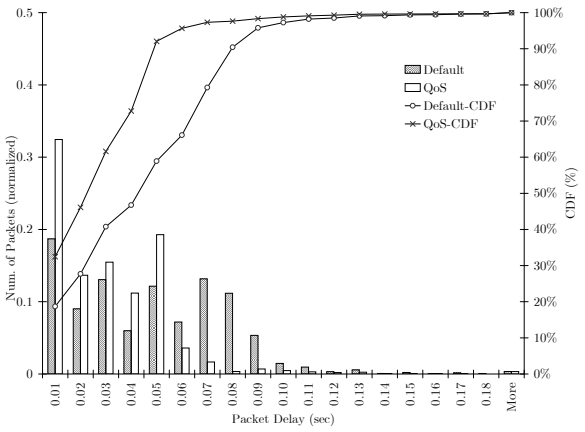


Figure 7: Histogram of Voice Packet Delay

the packet loss in the system at the cost of further delaying the voice packets. However we need to note that for the same amount of packet loss (e.g. 5%), the QoS-aware scheduler requires a smaller jitter buffer size (70 ms) as compared to the round robin scheduler (100 ms).

6 Conclusion

In this paper, we have presented a new QoS-aware channel scheduling technique for multi-radio/multi-channel wireless mesh networks. Through the use of traffic priority information carried in the packet header, we propose a new channel assignment scheduler that gives priority to VoIP traffic and still guarantees reasonable through of the non-priority TCP traffic. The main improvement of this technique is the replacement of the round robin scheduler by a QoS-aware scheduler that consider traffic priorities. This technique has been integrated to the Net-X platform and through testbed measurements we show the reduction in average end-to-end delay and network jitter of VoIP flows for the proposed QoS-aware channel scheduler.

As future works, we plan to investigate two issues. First, we plan to carry an extended study where different scenarios with greater number of nodes and different traffic demands are analyzed. In our current performance evaluation, we make use of static traffic priorities among flows. Therefore, our second work is to evaluate the assignment of dynamic priorities.

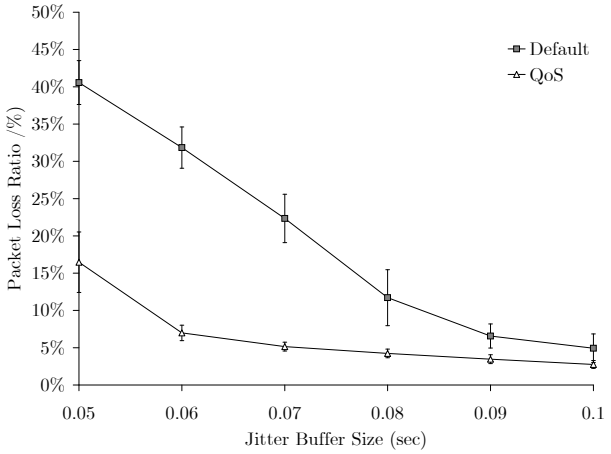


Figure 8: Packet Loss Ratio after the Jitter Buffer

7 Acknowledgments

This research is supported by the European Regional Development Fund through the Interreg IVB North Sea Region Project on "European Collaborative Innovation Centres for Broadband Media Services" (E-CLIC), and by National Science Foundation grant CNS 06-27074.

References

[1] iperf. URL: http://www.noc.ucf.edu/Tools/Iperf/.

[2] KAUMesh Testbed Wiki. URL: http://www.cs.kau.se/cs/prtp/pmwiki/pmwiki.php?n=Resources.MeshTestbed.

[3] Multi-generator (mgen). URL: http://cs.itd.nrl.navy.mil/work/mgen/.

[4] C. Chereddi and P. Kyasanur and N. H. Vaidya. Design and Implementation of a Multi-Channel Multi-Interface Network. In *Proc. of REALMAN*, 2006.

[5] J. Crichigno, M. Wu, and W. Shu. Protocols and architectures for channel assignment in wireless mesh networks. *Ad Hoc Networks*, 6:1051–1077, September 2008.

[6] T. Gleixner and D. Niehaus. Hrtimers and beyond: Transforming the linux time subsystems. In *Linux Symposium*, volume 1, pages 333–346, Ottawa, Canada, 2006.

[7] P. Kyasanur and N. H. Vaidya. Routing and link-layer protocols for multi-channel multi-interface ad hoc wireless networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 10(1):31–43, 2006.

[8] A. Mishra, V. Shrivastava, D. Agrawal, S. Banerjee, and S. Ganguly. Distributed channel management in uncoordinated wireless environments. In *Proc. of MobiCom'06*, pages 170–181, New York, NY, USA, 2006. ACM.

[9] Q. Ni. Performance analysis and enhancements for ieee 802.11e wireless networks. *Network, IEEE*, 19(4):21–27, 2005.

[10] K. Nichols, S. Blake, F. Baker, and D. Black. RFC 2474: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 headers, December 1998.

[11] S. Ganguly and V. Navda and K. Kim and A. Kashyap and D. Niculescu and R. Izmailov and S. Hong and S. Das. Performance optimizations for deploying VoIP services in mesh networks. *Selected Areas in Communications*, 24(11), 2006.

[12] T. B.-Y. Shen. Experiments on a multichannel multi-interface wireless mesh network. Master's thesis, University of Illinois at Urbana-Champaign, May 2008.

[13] B. Walke, S. Mangold, and L. Berlemann. *IEEE 802 Wireless Systems: Protocols, Multi-Hop Mesh/Relaying, Performance and Spectrum Coexistence*. John Wiley & Sons, Nov 2006.