In Proc. of European Wireless 2005, Nicosia, Cyprus, April 2005

A Performance Comparison of QoS Approaches for Ad Hoc Networks: 802.11e versus Distributed Resource Allocation

Emma Carlson[1], Christian Bettstetter[2], Christian Prehofer[2], Adam Wolisz[1]

[1] **Technische Universität Berlin, Telecommunication Networks Group, Berlin, Germany**
{carlson|wolisz}@tkn.tu-berlin.de

[2] **DoCoMo Euro-Labs, Future Networking Lab, Munich, Germany**
{bettstetter|prehofer}@docomolab-euro.com

***Abstract*: We compare two approaches for Quality of Service support in WLAN-based ad hoc networks. The first approach is to use per-packet priorities, according to the IEEE 802.11e standard. The second approach is to allocate radio resources on the path between source and destination, according to our protocol 'Distributed end-to-end Allocation of time slots for REal-time traffic' (DARE).**

Performance simulations show the following results: In case of low load, IEEE 802.11e has slightly lower end-to-end delay and higher packet loss rate, since it does not use any coordination among nodes for real-time packets. In case of medium load, DARE is superior in terms of jitter, delay, and packet loss. In case of high load, DARE clearly outperforms 802.11e. The results still hold if DARE has to repair the resource reservation path due to node failures.

1. Introduction

Applying the *Distributed Coordination Function* (DCF) of IEEE 802.11 in multi-hop wireless networks leads to uncertainties as to when a node can transmit over the next hop. These uncertainties sum up over multiple hops, hence throughput and end-to-end delay can suffer from large variations. This is especially crucial for real-time applications with demanding Quality of Service (QoS) requirements. To support a certain level of QoS, extensions or modifications to DCF are needed. Two basic approaches can be employed.

The first approach is to assign different *priority levels* to packets [21, 13, 17, 11, 18, 3]. As packets arrive at a node, it handles them according to their priority level. Packets with high priority (e.g., real-time packets) are transmitted prior to packets with low priority. The major issue is how to assign these priorities. This is typically done by defining different intervals for both the random backoff period and interframe period. For example, the contention-based access scheme in IEEE 802.11e, called *Enhanced Distributed Channel Access* (EDCA) [3], implements different packet priorities. Packets with high priority have low random backoff periods and low interframe periods. A node handles the packets in different queues, such that a packet from a queue with higher priority is sent before a packet from a queue with lower priority. In a single hop environment, EDCA offers better average delay and throughput than the usual DCF [15]. In a multi-hop environment, the throughput for a flow with high priority can be maintained stable, and the average end-to-end delay can be kept within bounds [6].

The second approach to support QoS is to *reserve resources* for a particular real-time traffic flow. For example, each node between source and destination allocates some dedicated time slots for this flow before the actual transmission starts. On the one hand, this removes uncertainties that come with a distributed random medium access. It thus has the potential to support applications demanding a non-varying end-to-end delay. On the other hand, such a reservation mechanism is typically much more complex than a priority mechanism. In particular, it adds signaling overhead to coordinate the nodes. All nodes between source and destination must agree in distributed manner on the reserved resources, the non-participating nodes must be informed so they abstain from transmission, and the reservation must be maintained and re-established when broken.

The goal of this paper is to compare a priority-based and a reservation-based QoS approach, in particular with respect to their end-to-end delay (average delay and jitter). As priority-based approach, we use EDCA of 802.11e. As reservation-based approach, we use the *Distributed end-to-end Allocation of time slots for REal-time traffic* (DARE) protocol, which was presented in our previous work [7, 8]. We show that DARE offers a more stable end-to-end delay with very low jitter, and it "guarantees" a certain delay of real-time packets even if the background traffic is very high and the reservation must be frequently re-established due to node failures.

This remainder of this paper is organized as follows. Sections 2 and 3 recall the functionality of DARE and EDCA, respectively. Section 4 describes the simulation model used for performance comparison. Section 5 presents and interprets the simulation results. Finally, Section 6 concludes and gives ideas for further research.

2. Distributed Resource Allocation with DARE

Most existing reservation mechanisms allocate resources in a single-hop fashion [10, 12, 14, 20, 19, 23, 16]. The DARE protocol, however, reserves resources on the entire path between source and destination before the actual data transmission can begin. This reservation is performed in a completely distributed manner. The following paragraphs summarize the basic protocol functionality. More detailed information can be found in [7, 8].

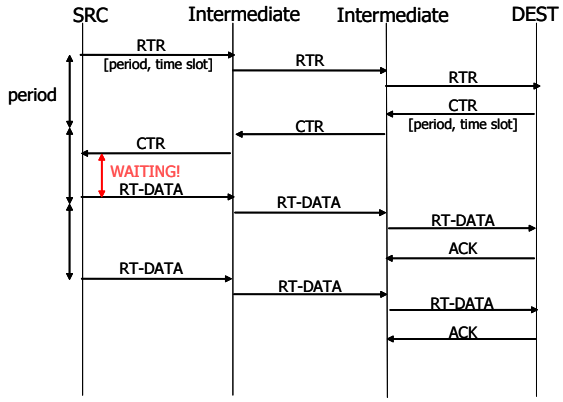


Figure 1: DARE: reservation setup and data transmission

Reservation Setup As shown in Figure 1, the source node sends a *Request-To-Reserve* (RTR) message containing the requested duration and periodicity of a time slot. This message is propagated via all intermediate nodes to the destination node. Each intermediate node forwards the RTR if it can accept the reservation request. The destination node responds using a *Clear-To-Reserve* (CTR) message, which propagates back to the source node and indicates to all intermediate nodes that the reservation request is accepted. As the source node receives the CTR, it can start the transmission of real-time packets in the following time slot. If the reservation request cannot be met by all intermediate nodes, some intermediate nodes will not receive the CTR and thus release their resources after a certain time period. In essence, we make 802.11 aware of TDMA-inspired reserved time slots, while retaining the usual DCF for non-real-time data.

Reservation Protection To protect the reservation path from interference, it is important that nodes located in the vicinity of the reservation path are also informed about the reserved time slots. Nodes that are directly adjacent to the reservation path are informed by overhearing the setup messages or real-time data packets. Unfortunately, for carrier sensing networks, the range where a node can interfere is much larger than the range where a node can receive [9]. This implies that nodes that do not overhear (cannot decode) the setup messages can still interfere, if they start a transmission directly before the reserved time slot. Therefore, the protection of the reservation is extended to a two-hop neighborhood. This is implemented in a way that adjacent nodes that are aware of the reservation but not participating inform non-aware nodes using the RTS/CTS exchange of non-real-time transmissions.

Data Transmission Upon completion of the reservation setup, the source node can transmit real-time packets during the reserved time slots without performing additional medium access control. Lost real-time packets are not retransmitted, since collisions are very unlikely.

Reservation Maintenance and Repair If the nodes switch on and off or move, the path between source and destination may break. Hence, we need a distributed scheme to handle such broken paths.

The task of the DARE protocol is to detect the broken path and pass this information from the MAC layer to the network layer. The routing protocol can then find a new path (route repair). Once this new path is found, the DARE protocol will repair the resource allocation.

For the DARE protocol to detect a broken path, we need to acknowledge every transmission of a real-time packet. Then, a node notices if its next-hop neighbor is no longer alive or moved out of reach. We have designed DARE to use implicit acknowledgments (iACK), i.e. each node overheads the real-time transmission of its subsequent node in the path in the next time slot. This can be done because we use periodic time slots. At the last hop, an explicit acknowledgment (eACK) is used, as the destination node does not forward the packet.

Upon detection of the broken path, DARE informs the routing protocol. Some routing protocols support both local and source-initiated route repair. In this case, DARE reserves the reservation accordingly: For a source-initiated route repair on the network layer, DARE performs source-initiated reservation repair on the MAC layer. For a local route repair, it performs a local reservation repair. During the repair procedure, the messages of the routing protocol are given priority, and real-time packets are buffered. Reservations that are no longer needed will time out after three unused time slots (initial simulations have shown that this is a reasonable value).

3. IEEE 802.11e

Before we explain the QoS functionality of 802.11e, let us briefly recall the basic functionality of the standard DCF [2]: Before a node is allowed to transmit it must sense the channel. If the channel is busy, the node backs off for a certain time period. If the channel is idle, the node waits a *Distributed Inter Frame Space* (DIFS) and continues to sense the channel. If the channel is still idle after this waiting period, the node transmits. If the channel is busy after this waiting period, the node backs off. The time period that a node has to back off upon a busy channel is determined by a random integer number and the physical layer parameter $aSlotTime$. A random integer number is chosen from a contention window [CW_{min}, CW_{max}], which acts as the starting value for a counter: The node decreases the counter by one if the channel is idle for a time period of $aSlotTime$. If the channel is busy during this period, the node freezes the counter until the channel is idle again. Once the counter reaches zero, the node transmits.

The Enhanced Distributed Channel Access (EDCA) of 802.11e [3] is an extension of this mechanism to include different priorities to packets. Packets of high priority should have a higher probability of gaining medium access. To do so, four *access categories* have been defined: voice, video, best effort, and background traffic. Voice has the highest and background traffic the lowest priority. Each category has its own queue within a

Access Category	voice	video	best effort	background	DCF
CW_{min}	7	15	31	31	31
CW_{max}	15	31	1023	1023	1023
AIFSN	2	2	3	7	2

Table 1: Back off and AIFSN values for EDCA and DCF

node. Packets from the queue with the highest priority are transmitted first. When this queue is empty, packets from the second highest priority queue are transmitted, and so on. Furthermore, each category has a different contention window and backoff times.

Table 1 shows the minimum and maximum values of the contention window (CW_{min}, CW_{max}). Voice packets have the lowest backoff interval (from 7 to 15); best effort and background packets have the same backoff interval as usual DCF (from 31 to 1023).

The time period that a node has to sense a channel to be idle before it is allowed to transmit, is called *Arbitrary Inter Frame Space* (AIFS) in EDCA. It is determined according to

$$\text{AIFS} = \text{AIFSN} \cdot aSlotTime + aSIFSTime\,,$$

where the number AIFSN is defined by the access category (see Table 1), and $aSIFSTime$ is a parameter of the physical layer. By assigning packets with high priority a small AIFSN, the waiting time before transmission becomes smaller.

4. Simulation Model

In our simulation scenario, 100 nodes are distributed on a square area using a uniform random distribution. All nodes transmit with such power that they have a communication range of about 200 m, using a radio model from the Lucent WaveLan interface card. To guarantee that the resulting network is connected with high probability (larger than 99 %), the length of the area is set to 700 m [4]. The multi-hop paths are found using the Ad hoc On Demand Distance Vector (AODV) routing protocol. All nodes transmit data packets with 1 Mbps, which is the basic data rate channel of IEEE 802.11.

We generate one real-time flow in this network. The destination and source nodes are randomly chosen from the 100 nodes. Every 0.1 s, the source node generates a real-time packet of size 512 bytes on the application layer. When the packet is transmitted on the physical layer, it has a size of 600 bytes, which includes UDP, IP, MAC, and physical-layer header and preambles. Thus, the transmission using the 1 Mbps channel takes $\tau = 4.8$ ms.

The remaining 98 nodes transmit non-real-time (nrt) traffic. They transmit packets with 512 bytes size with exponentially distributed inter-arrival times. We compare the outcome of three different load values for this background traffic: $\lambda = 5$ kbps, 10 kbps, and 100 kbps per node. All nrt-nodes transmit to the destination of the real-time transmission, hence the randomly chosen destination node can be interpreted as an access point or gateway of a multi-hop access network.

Using DARE, we must reserve time slots of length $\tau = 4.8$ ms every 0.1 s. Intermediate nodes need both a receive and transmit slot, thus 9.6 ms is reserved. The simulations of IEEE 802.11e are performed with the highest priority (voice) for the real-time flow and lowest priority for the background traffic (see Table 1). We set $aSlotTime = 20\,\mu$s and $aSIFSTime = 10\,\mu$s. For more parameters of the EDCA simulation, see [22].

To simulate network dynamics, every nrt node switches off after some random time, which is sampled from a negative exponential distribution with a given expected value $E\{T_{\text{on}}\}$. It switches on again after another random time, sampled from the same distribution with the expected value $E\{T_{\text{off}}\}$, and so on. For simplicity, we set $E\{T_{\text{on}}\} = E\{T_{\text{off}}\}$ and call this parameter μ.

We use the NS-2 simulator [1] with a simulation time of 3600 s. We simulate 50 random scenarios and then take the average of the performance values.

5. Results

5.1 Average Packet Delay and Packet Loss Rates

Figure 2 shows the average end-to-end delay of real-time packets over μ for three different values of the background traffic ($\lambda = 5$ kbps, 10 kbps, and 100 kbps per nrt node). Using EDCA, the background traffic has a huge impact on the delay. With 100 kbps load, the average delay is almost three times as large as with 5 kbps. The delay using DARE is independent of the background traffic. For both protocols, if nodes switch on and off more frequently (low μ) more delay is introduced, which is caused by more frequently performed re-routing procedures.

Let us now compare the average delay values of the two protocols in a qualitative manner. If much background traffic is transmitted, EDCA suffers from a much higher delay than DARE. For medium background traffic load, both protocols show similar delay values. Only if the background load is very low, EDCA is superior to DARE. The reason for the latter result is that, in our simulations, EDCA does not employ any exchange of RTS/CTS messages. This lack of coordination, however, can lead to high packet losses if small CW intervals are used, as it is then very likely that the backoff timer of two or more nodes expires at the same moment.

Thus, we also analyze the packet loss rate, which is shown in Figure 3. As can be seen, DARE has a lower packet loss rate for all values of the on/off time and all background loads. Packet losses with DARE are mainly

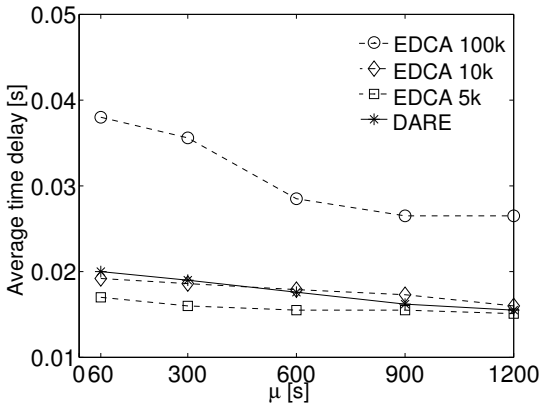


Figure 2: Average end-to-end delay of real-time packets

caused by interference from nodes that are more than one hop away from the real-time transmission path. Some of this interference is avoided, as discussed in Section 2, but not all. As retransmission of lost packets makes often little sense in real-time applications, the high packet loss rates of EDCA imply that data not to reach the final destination node.

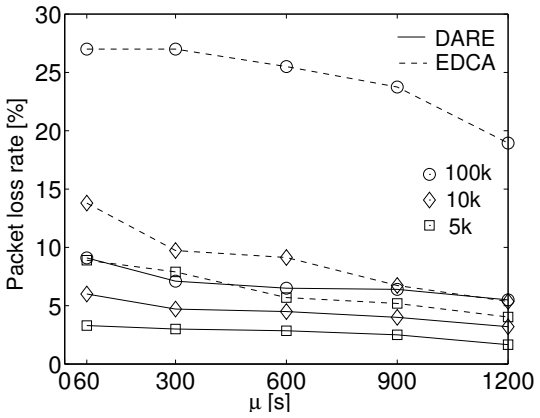


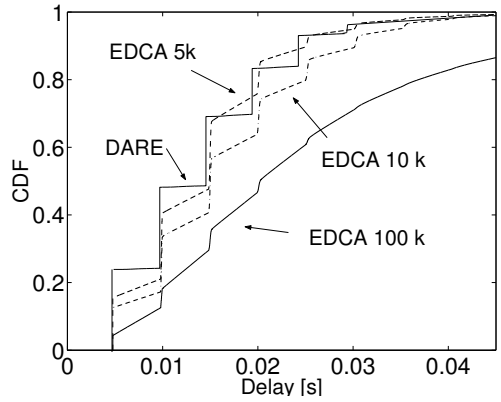
Figure 3: Packet loss rate of real-time packets

In summary, if the background traffic load is small, EDCA has slightly lower average time delay but higher packet loss rate. For high background loads, however, DARE has a much lower average time delay and lower packet loss rate. We can also say that EDCA has a trade-off between the packet loss rates and the time delay, which does not exist for DARE.

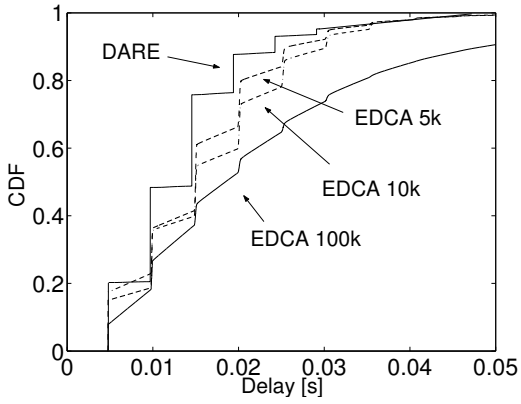
5.2 Distribution of Packet Delay

We now analyze the delay of real-time packets in more detail. Clearly, the end-to-end delay of a packet depends on the number of hops h on the path between the source and destination node. In our scenario, we pick the source and destination node randomly on the given area. Hence, the number of hops is a random variable. Simulations with the given parameters have shown that the value range of h is between 1 and 6, where most source-destination pairs have $h = 2$ and 3 (see also [5]).

Using DARE, the transmission duration is $\tau = 4.8\,\text{ms}$. Since no random access is needed, the transmission via one hop takes about this time. The total end-to-end delay is the sum of the delays of each hop. If the path between source and destination remains unbroken during transmission, the end-to-end delay will be $h\tau$. If the path breaks, the delay can also be higher than $h\tau$ because packets will be buffered during the repair process. This behavior is well reflected in the DARE simulations results shown in Figure 4.



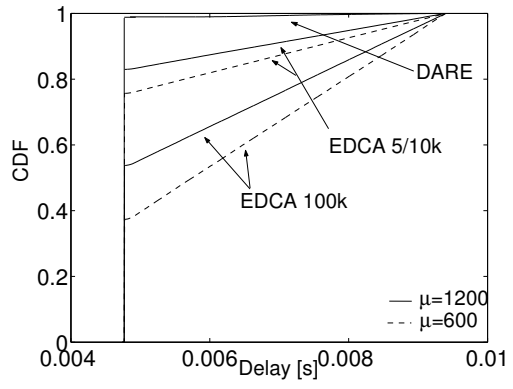
(a) $\mu = 600\,\text{s}$



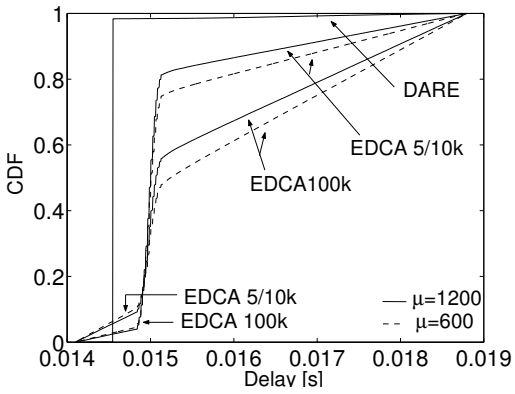
(b) $\mu = 900\,\text{s}$

Figure 4: End-to-end delay of real-time packets

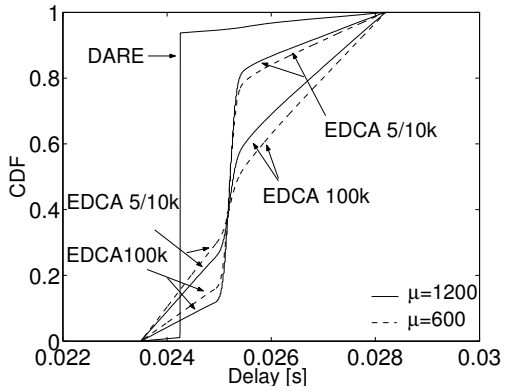
The cumulative distribution function (cdf) of the end-to-end delay is a step function with step width τ. The delay cdf for $\mu = 600\,\text{s}$ can be interpreted as follows: About $20\,\%$ of all real-time packets are transmitted via only one hop ($h = 1$), hence their delay is $5\,\text{ms}$. About $50\,\%$ of all real-time packets experience a delay that is at most $10\,\text{ms}$, about $70\,\%$ have at most $15\,\text{ms}$, and so on. The result is similar for $\mu = 900\,\text{s}$ with slightly increased occurrence of lower delays caused by less frequent path breaks. If the number of hops is known, the packet delay is quite predicable; a sudden change of the packet delay only occur for the first packets upon repair of a broken path. In total, the experienced jitter is very low. As we decrease the background load to $\lambda = 10\,\text{kbps}$ and



(a) Packets after $h = 1$ hop



(b) Packets after $h = 3$ hops



(c) Packets after $h = 5$ hops

Figure 5: Delay of real-time packets after h hops. The results for $\lambda = 5$ kbps and 10 kbps are very similar.

finally $\lambda = 5$ kbps, the performance of EDCA improves but is still inferior to DARE (except for some particular delay bounds around 20 ms if $\mu = 600$ s). The delay behavior of packets using EDCA is completely different. Due to the random nature of the medium access in EDCA, the delay can now take values from a continuous value range. Hence, the cdf is no longer a step function. For much background traffic, EDCA achieves a much lower performance than DARE. In the simulation with $\lambda = 100$ kbps and $\mu = 600$ s, for instance, a delay below 20 ms is achieved by about 50 % of all packets, compared to more than 80 % using DARE. Also the jitter in the delay using EDCA is higher and increases with increasing number of hops.

In a last experiment, we are now interested in the delay for all real-time packets at a given hop distance h from the sender. The ideal delay value at hop distance h would be $h\tau$ with $\tau = 4.8$ ms. Figure 5 shows the measured results for $h = 1$, 3, and 5. Using DARE, almost 100 % of the real-time packets arrive at the first hop within 4.8 ms and at the third hop within 14.4 ms. About 95 % of the packets arrive at the fifth hop within 24.2 ms. Hence, almost all packets arrive at a well-defined time, thus no jitter occurs. Using EDCA, there are some few packets that arrive faster than with DARE. The majority of packets, however, arrives later. For example, with $\lambda = 100$ kbps background load and $\mu = 600$ s, the delay of 80 % of the packets is still above 24.2 ms at the fifth hop.

6. Conclusions

This paper investigated the differences between two QoS-enabling MAC protocols applied in WLAN-based wireless ad hoc networks. We compared the end-to-end delay of the standardized, priority-based 802.11e EDCA and our reservation-based DARE protocol. The simulation-based analysis was done in a random network where nodes switch on and off, hence forcing the reservation to break and re-establish.

In conclusion, DARE has the capability to give real-time flows a more stable end-to-end delay with very low jitter. A major advantage is its capability to offer a guaranteed delay of real-time packers even if the background traffic is very high. This is especially important to provide QoS to important traffic flows in overload scenarios. EDCA can offer a bounded average end-to-end delay, but the delay jitter is much larger, especially for a high number of hops and much background traffic. Even in scenarios where paths break often and the reservation

repeatedly has to be re-established, DARE outperforms the less complex priority mechanism EDCA both in delay and packet losses.

Ongoing research activities include the support of multiple reservation flows and a thorough analysis of the impact that DARE has on the non-real-time traffic.

REFERENCES

[1] Network Simulator 2. www.isi.edu/nsnam/ns/.

[2] IEEE 802.11. "Standard 802.11: Wireless LAN Medium Access Control (MAC) and Physical layer (PHY) Specifications". 1999.

[3] IEEE 802.11e WG. "Draft supplement to Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: MAC Enhancements for Quality of Service (QoS), IEEE Std 802.11e/D9.0". August 2004.

[4] C. Bettstetter. "On the Connectivity of Wireless Multihop Networks with Homogeneous and Inhomogeneous Range Assignment". In *Proc. IEEE Vehicular Technology Conf. (VTC)*, Vancouver, Canada, September 2002.

[5] C. Bettstetter and J. Eberspächer. "Hop Distances in Homogeneous Ad Hoc Networks". In *Proc. IEEE Vehicular Technology Conf. (VTC)*, Jeju, Korea, April 2003.

[6] C. T. Calafate, P. Manzoni, and M. P. Malumbres. "Assessing the effectiveness of IEEE 802.11e in multi-hop mobile network environments". In *Proc. IEEE MASCOTS*, Volendam, Netherlands, October 2004.

[7] E. Carlson, C. Bettstetter, H. Karl, C. Prehofer, and A. Wolisz. "Distributed Maintenance of Resource Reservation Paths in Multihop 802.11 Networks". In *Proc. IEEE Vehicular Technology Conf. (VTC)*, Los Angeles, USA, September 2004.

[8] E. Carlson, H. Karl, A. Wolisz, and C. Prehofer. "Distributed Allocation of Time Slots for Real-time Traffic in a Wireless Multi-hop Network". In *Proc. European Wireless*, Barcelona, Spain, February 2004.

[9] S. Desilva and R. V. Boppana. On the Impact of Noise Sensitivity on Transport Layer Performance in 802.11 based Ad Hoc Networks. In *Proc. IEEE ICC*, Paris, France, June 2004.

[10] G. R. Hiertz, J. Habetha, P. May, E. Weiß, R. Bagul, and S. Mangold. "A Decentralized Reservation Scheme for IEEE 802.11 Ad Hoc Networks". In *Proc. IEEE PIMRC*, Beijing, China, September 2003.

[11] L. Jacob, R. Radhakrishna Pillai, and B. Prabhakaran. "MAC Protocol Enhancements and a Distributed Scheduler for QoS Guarantees over the IEEE 802.11 Wireless LAN". In *Proc. IEEE Vehicular Technology Conf. (VTC)*, Boston, USA, September 2002.

[12] S. Jiang, J. Rao, D. He, X. Ling, and C. C. Ko. "A Simple Distributed PRMA for MANETs". *IEEE Trans. on Vehicular Technology*, March 2002.

[13] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly. "Distributed Multi-Hop Scheduling and Medium Access with Delay and Throughput Constraints". In *Proc. ACM Mobicom*, Rome, Italy, July 2001.

[14] C. H. R. Lin and M. Gerla. "Asynchronous Multimedia Multihop Wireless Networks". In *Proc. IEEE Infocom*, Kobe, Japan, April 1997.

[15] S. Mangold, S. Choi, P. May, O. Klein, G. Hiertz, and L. Stibor. "IEEE802.11e Wireless LAN for Quality of Service". In *Proc. European Wireless*, Florence, Italy, Feb 2002.

[16] M. K. Marina, G. D. Kondylis, and U. C. Kozat. "RBRP: a Robust Broadcast Reservation Protocol for Mobile Ad Hoc Networks". In *Proc. IEEE ICC*, Helsinki, Finland, June 2001.

[17] Q. Qiang, L. Jacob, R. Radhakrishna Pillai, and B. Prabhakaran. "MAC Protocol Enhancements for QoS Guarantee and Fairness over the IEEE 802.11 Wireless LAN". In *Proc. Conf. Computer Commun. Net. (ICCNC)*, Miami, USA, October 2002.

[18] D. Qiao and K. G. Shin. "Achieving Efficient Channel Utilization and Weighted Fairness for Data Communications in IEEE 802.11 WLAN under the DCF". In *Proc. IEEE Intl. Workshop on Quality of Service*, Miami, USA, May 2002.

[19] S.-T. Sheu, T.-F. Sheu, C.-C. Wu, and J.-Y. Luo. "Desing and Implementation of a Reservation-based MAC Protocol for Voice/Data over IEEE802.11 Ad Hoc Wireless Networks". In *Proc. IEEE ICC*, Helsinki, Finland, June 2001.

[20] J. L. Sobrinho and A. S. Krishnakumar. "Real-time Traffic over the IEEE 802.11 Medium Access Control Layer". *Bell Labs Technical Journal*, pages 172–187, 1996.

[21] N. H. Vaidya, P. Bahl, and S. Gupta. "Distributed Fair Scheduling in Wireless LAN". In *Proc. ACM Mobicom*, August 2000.

[22] S. Wiethoelter and C. Hoene. "Design and Verification of an IEEE 802.11e EDCF Simulation Model in ns-2.26". Technical Report TKN-03-019, Telecommunication Networks Group, Technische Universität Berlin, November 2003.

[23] Z. Ying, A. L. Ananda, and L. Jacob. "A QoS Enabled MAC Protocol for Multi-Hop Ad Hoc Wireless Networks". In *Proc. IEEE Intl. Conf. on Performance, Computing, and Communications (IPCCC)*, Phoenix, USA, April 2003.