A Distributed End-to-End Reservation Protocol for IEEE 802.11-based Wireless Mesh Networks

Emma Carlson, Christian Prehofer, Christian Bettstetter, *Member, IEEE,*
Holger Karl, *Member, IEEE,* Adam Wolisz, *Senior Member, IEEE*

***Abstract*— This article presents an end-to-end reservation protocol for Quality of Service (QoS) support in the MAC layer of wireless multihop mesh networks. It reserves periodically repeating time slots for QoS-demanding applications, while retaining the Distributed Coordination Function (DCF) for best-effort applications. The key features of the new protocol, called 'Distributed end-to-end Allocation of time slots for REal-time traffic' (DARE), are distributed setup, interference protection, and scheduling of real-time data packets, as well as the repair of broken reservations and the release of unused reservations.**

A simulation-based performance study compares the delay and throughput of DARE with those of DCF and the priority-based Enhanced Distributed Channel Access (EDCA) used in IEEE 802.11e. In contrast to DCF and EDCA, DARE has a low, non-varying delay and a constant throughput for each reserved flow.

***Index Terms*— Medium access control, QoS, reservation mechanism, CSMA, IEEE 802.11e, mesh networks.**

I. Introduction

WIRELESS local area networks (WLANs) based on the IEEE standard 802.11 have become extremely popular. More than ever, people use WLANs for wireless Internet access, Voice over IP, file sharing, and other applications. As a reaction to this success, there is an increasing interest in enhancing the WLAN standard. The 802.11 working group has thus defined several new functionalities on both the physical and data link layers. Besides standardization of higher-rate transmission schemes [1] and better security functions [2], one of the most recent activities aims at extending the single-hop paradigm of current 802.11 technologies to a multihop paradigm. As illustrated in Fig. 1, the idea is to enable each network node to route traffic coming from other nodes. A packet sent by a device is relayed from one node to another, hop by hop, until it reaches an Internet access point or the intended receiver. This concept results in a mesh-like topology, as opposed to the star-like topology of conventional WLANs.

The major goal of such mesh networks is to improve the radio coverage, while assuring simple configuration and high reliability. Besides implementing automatic topology learning and dynamic path configuration, revision of the medium access control (MAC) layer is essential, since the basic random access protocol of 802.11 — the Distributed Coordination Function (DCF) [3] — is not well suited for mesh networks [4].

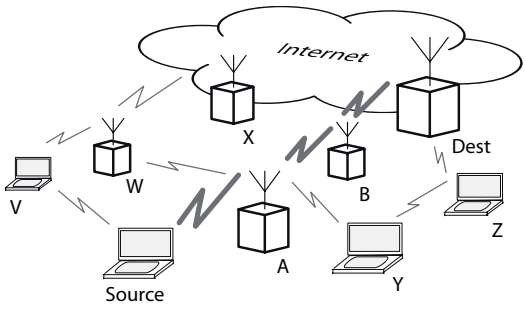


Fig. 1. Illustration of a mesh network.

Mesh networks are expected to handle various real-time applications (e.g., Voice over IP) in addition to the classical best-effort applications (e.g., email, web browsing). These real-time applications require low, stable packet delay and constant throughput. Such Quality of Service (QoS) support, however, is not included in the original DCF. This is true for single-hop communication and even more for multihop communication, where the negative effects accumulate if we cascade multiple non-synchronized, autonomously operating hops.

A possible improvement is to give packets originating from real-time applications a higher *priority* when accessing the shared channel. This concept has been standardized in IEEE 802.11e, introducing the Enhanced Distributed Channel Access (EDCA) [5] protocol. It defines four different traffic categories: voice, video, best effort, and background.

The alternative QoS approach, motivated in principle by circuit switching, is to perform an end-to-end *reservation* for each real-time flow. While this approach potentially assures end-to-end QoS, its drawback is the need for resource management in each node and inter-node signaling. Neither is widely available in IEEE 802.11 networks. This lack has been our motivation to develop a protocol in this domain, which is well suited for multihop mesh networks. Its design and performance analysis is presented in this article. The protocol operates in the MAC layer, where it reserves periodically occurring time slots in the nodes in a completely distributed manner; hence, it is called *Distributed end-to-end Allocation of time slots for REal-time traffic* (DARE). To be more specific, before a real-time transmission can begin, DARE reserves time slots in all

Manuscript received October 1, 2005; revised April 21, 2006. The work of E. Carlson was supported by DoCoMo Euro-Labs, Munich, Germany.

E. Carlson is with the Telecommunication Networks Group, Berlin University of Technology, 10587 Berlin, Germany.

C. Prehofer is with Nokia Research, 00180 Helsinki, Finland. Research was carried out at DoCoMo Euro-Labs, 80687 Munich, Germany.

C. Bettstetter is with the Mobile Systems Group, University of Klagenfurt, 9020 Klagenfurt, Austria.

H. Karl is with the Computer Networks Group, University of Paderborn, 33098 Paderborn, Germany.

A. Wolisz is with the Telecommunication Networks Group, Berlin University of Technology, 10587 Berlin, Germany.

nodes along the route between the source node of a real-time flow and its final destination node. It then schedules the real-time data packets between the nodes, transmitting them in the reserved time slots. The protocol protects the allocated time slots from interference by informing nodes located near the real-time path. The adjacent nodes will thus abstain from transmitting during the reserved time slots. Finally, DARE also handles the repair of broken reservation paths and the release of invalidated reservations. In essence, the protocol extends the spatial reservation concept of 802.11 — achieved by the exchange of Request-to-Send (RTS) and Clear-to-Send (CTS) messages — to a multihop, end-to-end perspective.

A wireless mesh network will use DCF and DARE in parallel. Data packets coming from non-real-time applications use the Carrier Sense Multiple Access (CSMA) approach of DCF, either with or without the exchange of RTS/CTS messages. Data packets from real-time applications, however, use DARE and are transmitted during the reserved time slots. The medium access results in a combination of CSMA and Time Division Multiple Access. The routes between source and the destination of the real-time traffic are found by a wireless routing protocol, e.g., the Ad hoc On-Demand Distance Vector Routing (AODV) protocol [6].

This article introduces the DARE protocol and analyzes its performance, compared to standard DCF and priority-based QoS with EDCA. Section II explains in detail the DARE protocol with its different functional building blocks. Section III presents a simulation-based performance analysis of DARE, mainly evaluating end-to-end delay, jitter, and throughput in comparison to DCF and EDCA. We show that, in contrast to DCF and EDCA, DARE has a low, non-varying average delay and a constant throughput, even at high network load. Section IV describes related work, and finally, Section V concludes. Appendix A gives a brief overview of the IEEE DCF and EDCA standards and their parameters.

II. DARE Protocol

We consider a wireless mesh network with nodes capable of relaying traffic. Each node has implemented the standard DCF and the DARE protocol. A real-time flow consists of a periodic transmission of a fixed-size packet; different flows can have packets of different size and periodicity. A time slot is defined as the period of time that a node needs for either transmitting or receiving a given real-time packet. All nodes have synchronized, non-drifting clocks.

The DARE protocol [7]–[10] can be described by five functional building blocks: reservation setup, real-time data transmission, reservation protection, reservation repair, and reservation release. The following sections explain these building blocks in detail. Hereby, we use the scenario shown in Fig. 1: The source node wants to transmit real-time data packets to the destination node; nodes A and B serve as relays.

A. Reservation Setup: Basic Scheme

The task of the reservation setup is to reserve time resources in the nodes along a path between a source and a destination node. We first describe the reservation setup when there is only a single path; later in this article we address the case where multiple reservations exist simultaneously.

The resource reservation is initiated by the source node. As shown in Fig. 2, it sends a *Request-to-Reserve* (RTR) message, which includes the requested duration Δ and periodicity τ of a time slot as well as the address of the destination node. The message is sent to the next node on the path toward the destination node. If this node can fulfill the reservation request (see Section II-C for details), it makes an entry in a reservation table. The status of this entry is set to *preliminary*. It then forwards the RTR to the next hop. In this way, the message propagates, hop-by-hop, through the entire path via all intermediate nodes to the destination node. It indicates to all these nodes how often and for how long they must be available for real-time transmissions.

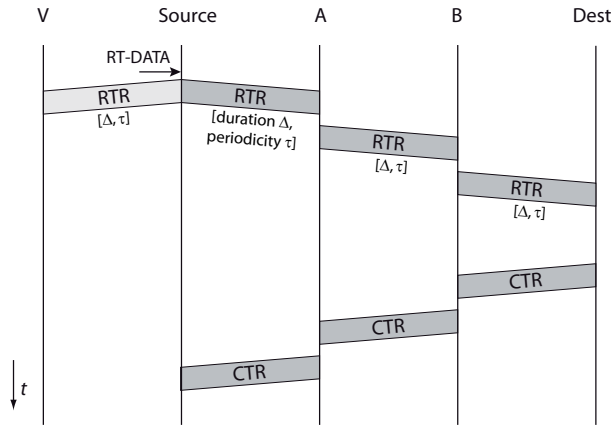


Fig. 2. Reservation setup. The message sequence chart corresponds to the mesh network of Fig. 1. Nodes W, X, Y, and Z are not shown for simplicity.

The destination node generates a *Clear-to-Reserve* (CTR) message. This message confirms the reservation request and travels along the same path back to the source indicating to the intermediate nodes that the reservation request is accepted by all nodes. The arrival of a CTR changes the status of the reservation table entries from *preliminary* to *fixed*. Upon receiving the CTR, the source node can start transmission of real-time traffic in the upcoming reserved time slots.

If a node cannot fulfill a reservation request, it does not forward the reservation message. This means that preceding nodes will not receive a CTR, and the reservation status will never be fixed. Although the preliminary status does not mean that a time slot is fully reserved, it can block other reservation requests occurring simultaneously. This preliminary reservation will be released after some time period if the end-to-end reservation is unsuccessful. A time-out function is used for this purpose. When a node sends the RTR to the next node in the path, it starts an RTR timer. If no CTR is received before the timer expires, the reservation is released. The duration of the RTR timer is a design parameter. Ideally, it can be changed according to traffic requirements or network topology or adapted after each unsuccessful reservation setup.

Nodes adjacent to the reservation path also receive the RTR and CTR messages, being thus informed about the reserved time slots. In essence, we extend the spatial reservation con-

cept of DCF to cover the whole multihop path. The structure of the two setup messages RTR and CTR is very similar to that of RTS and CTS messages, as is the need for these messages to be observed by adjacent nodes. The major extension is that RTR and CTR contain Δ and τ. These values are given to the MAC layer from the application layer. Both signaling messages also contain information about slots reserved for the same flow at other nodes of the same path. For medium access control of the RTR and CTR messages themselves, we use the standard DCF, where the messages are transmitted within the waiting time of a short inter-frame space (SIFS).

The routes between source and destination are found by some wireless routing protocol (e.g., AODV [6]). DARE requires the routing protocol to provide symmetric routes between source and destination. Mechanisms like symmetric route pinning [11] can be used to assure this. In principle, asymmetric routes for data flows can be supported; only the CTR has to travel back along the RTR's route.

B. Real-Time Data Transmission

Once the reservation is set up, the source node can transmit its real-time packets during the reserved time slots. Fig. 3 shows a message sequence chart of a real-time data transmission. Evidently, for one real-time flow, each relay node must reserve one time slot for receiving ("receive slot") and one slot for transmitting ("transmit slot"), both of duration Δ.

Due to the spatial reservation concept, nodes near a reserved path abstain from transmission during the reserved time slots. Hence, the possibility of interference is minimized.

DARE does not retransmit lost packets, since the delay constraints of real-time applications are typically much stricter than the packet loss constraints. Furthermore, we expect collisions to be in any case rare due to minimized interference.

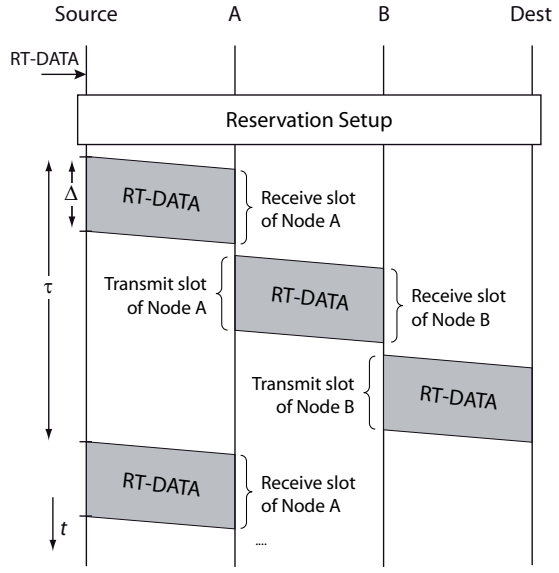


Fig. 3. Real-time data transmission.

C. Reservation Setup: Multiple Reservations

The reservation setup is straightforward if no node in the path already holds a reservation. If some nodes, however, are already committed to other DARE reservations, an overlapping of time slots might occur. In case of such conflicting reservations, a relay node can re-schedule (shift in time) its own, newly to be reserved time slot to reject as few reservation requests as possible, hence increasing the number of supported real-time flows.

Upon the arrival of an RTR, a node first checks whether the requested receive slot is conflicting (overlapping) with already existing reservations of the node. If the receive slot is conflicting, it transmits an *Update-Transmit-Reservation* (UTR) message back to the node that proposes the receive slot. It suggests at least one time slot suitable for reception. If the new slot can be fulfilled, preliminary reservations are released and a new RTR is generated.

If the receive slot is not conflicting, the node checks whether the transmit time slot is appropriate. If the transmit slot conflicts with existing slot reservations the node performs a re-scheduling taking different periodicities into account. Using the greatest common divider for all periodicities already accepted and the requested one, the node can find a suitable shift. Periodicities that do not have any common divider (e.g. prime numbers) are not allowed.

If a receive or a transmit slot cannot be scheduled, the reservation request is rejected and the flow is blocked.

D. Reservation Protection

Transmissions in the real-time path must be protected against interference. For this purpose, we apply a spatial reservation concept: nodes located close to the real-time path are informed about the reservation; they then abstain from transmitting during the reserved slots.

A basic level of protection is already achieved in the reservation setup phase, where nodes located in the transmission range of the real-time path overhear and obey RTR and CTR messages. In addition, reservation information is also contained in the header of real-time data packets and acknowledgments. This information consists again of the time slot duration Δ and periodicity τ. This informs nodes that did not overhear the reservation setup phase, for example, because they have switched on after the setup took place.

To achieve a higher level of spatial reservation, a node does not only avoid slots of its direct neighbors but also slots of nodes further backward and upward in the reservation path. To do so, the reservations of nodes up to two hops backward in the reservation path are piggy-backed. A node that overhears this information avoids three slots: The actual receive slot, the preceding receive slot of the node transmitting the message, and the receive slot of the node preceding the transmitting node. For instance, a node adjacent to the destination node in Fig. 3 avoids the receive slots of the destination, node B, and node A. If a conflict of a receive slot is discovered, the node sends an UTR to the respective node. If possible, a node also avoids time slots of nodes upward in the reservation path. It learns these time slots by overhearing the forwarded real-time packets (see Section II-E).

For a node to overhear messages not addressed to itself but to another node, it must operate in promiscuous mode; this comes at the price of higher energy consumption [12].

Another question with respect to reservation protection is how far the reservation information should be propagated. Nodes that are not located in the transmission range of the reservation path but one hop further away could cause interference as well, but might not be able to decode messages sent in the path and are thus not informed about reserved slots. The authors' publication [9] examined a protection technique in which all nodes that are two hops away from a reserved path are informed. We found that these reservations are hard to maintain and come at the cost of much explicit signaling. As most real-time traffic is robust to some packet losses and since additional signaling and the resulting lower spatial reuse factor can drain the capacity of a network, we do not use such two-hop protection in the DARE protocol. However, if the consequence is that two reservations are set up such that the slots overlap, both are released after a certain number of unsuccessful transmissions (see Section II-F) and the source re-initiates the setup.

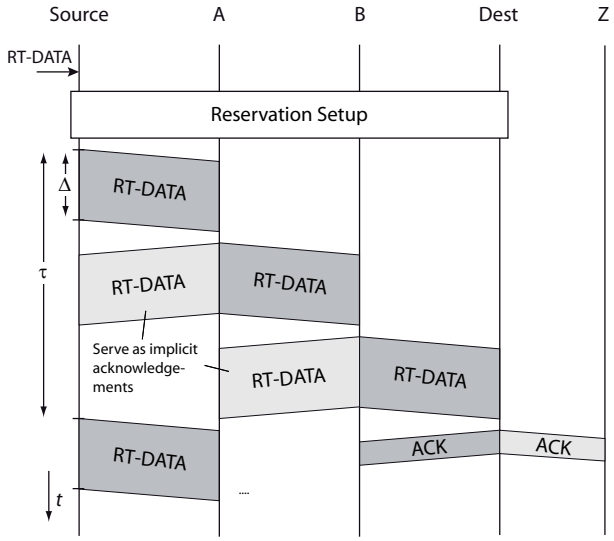


Fig. 4. Real-time data transmission and acknowledgment.

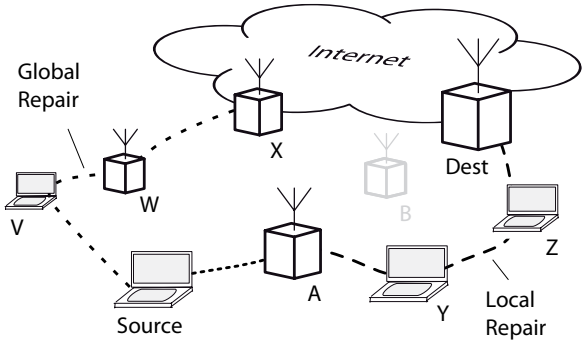


Fig. 5. Reservation repair scenario. Node B switches off. The routing protocol finds a new route, either using local or global repair. The new route is then reserved by the DARE protocol.

E. Reservation Repair

An established reservation path might break during the real-time transmission if the network topology changes. Such changes might occur if nodes switch off or fail or if the channel conditions change. Clearly, such path breaks must be repaired for the real-time transmission to continue. To initiate a path repair, the node preceding the "hole" in the path must notice the broken link. In the following, we discuss how nodes *detect* and how they *repair* a broken reservation path.

For a node to detect a broken link, the transmission of real-time packets must be acknowledged.[1] In DARE, such acknowledgments are achieved *implicitly*: if a node sends a real-time packet to the next node of the path, this transmission is acknowledged in such a way that the node overhears the next node's transmission of this packet onward (Fig. 4). Such implicit acknowledgments do not cause any signaling overhead. Only at the very last hop, where the destination node does not forward any data, does it acknowledge the reception of real-time packets in an *explicit* manner (see Fig. 4: ACK). The acknowledgment causes a small signaling overhead but has the advantage of informing nodes in the neighborhood of the destination node about the reservation, hence contributing to the protection of the reservation against interference. In both cases, if a node can no longer reach its subsequent node in the path, i.e., it does not receive acknowledgments, it assumes a broken link. Fig. 5 gives an example: node B switches off, and node A detects its link failure to node B.

The subsequent path repair is done in two steps: route repair and reservation repair. First, the MAC layer indicates the link break to the network layer. As in standard DCF, this event triggers the routing protocol to update the route. After the routing protocol has repaired the route between source and destination, the DARE protocol repairs the reservation at the MAC layer. Depending on the routing protocol, there are two options for reservation repair in the MAC layer (Fig. 5):

- *Local repair.* If the network route was repaired locally, DARE does so as well, which is effective and low in signaling. All nodes up to the node that detects the link break can keep their reservations as neither the period nor the time slot size has changed. Each node that is within range of the detecting node already has a reservation entry for this slot. Such nodes are potential candidates to act as new relay nodes. In the example, the link between the source and node A is maintained; the path between A and the destination is repaired locally via nodes Y and Z.
- *Global repair.* If the route repair was initiated by the source, the new route could be totally different from the old route. DARE releases all old reservations between source and destination (e.g., node A) and requests a new reservation via node V, W, and X to the Internet.

Finally, note that any node can force a reservation to be broken so that a route update procedure is initiated.

F. Reservation Release

Once a real-time flow has finished, all nodes belonging to this flow's reserved path should release their corresponding reservations. Similarly, if a reservation path breaks and a new

[1]We emphasize that these ACKs only serve in path maintenance; they are *not* used for retransmissions and are thus less critical in time or dependability.

one is established, the nodes of the old path should release their reservations, since they are no longer needed.

To release unused reservations, DARE employs a time-out. If a node does not receive or overhear any real-time data packet for a number of successive slots, it will release all reserved slots for this flow. The same release rule holds for nodes located adjacently to the reservation path, if they do not overhear real-time packets during their reserved slots.

Note that for some applications where silent periods occur, reservations might be falsely released. To avoid this, the source node is allowed to transmit dummy packets or the release time-out value could be increased during path setup.

III. Performance Analysis

A. Simulation Setup, Parameters, and Metrics

The DARE protocol is simulated by extending the well-known simulation tool NS-2 [13]. We use 400 mesh nodes, which are randomly uniformly located in a $2 \times 2\,\mathrm{km}^2$ area. The mesh network is connected to a wired network by four Internet gateways ("access points"), whose positions are given in Fig. 6. Ordinary nodes can switch on or off, modeling their participation in the mesh network. Both the on and off periods are modeled by exponentially distributed random variables with the same mean value μ. Nodes originating a flow or access points never switch off.

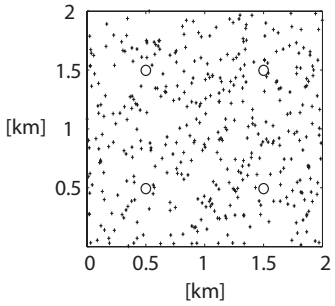
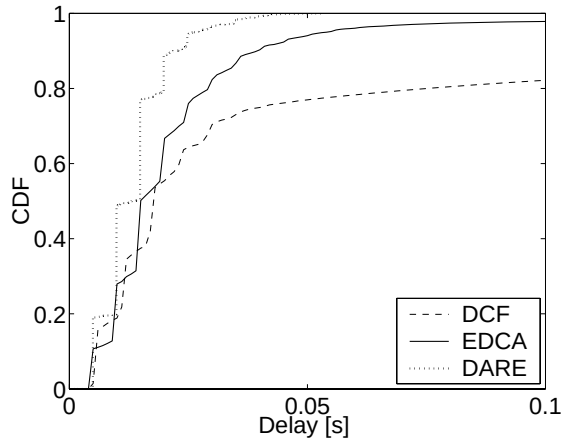


Fig. 6. Simulation setup: 4 gateways (o), 400 randomly located nodes (+).

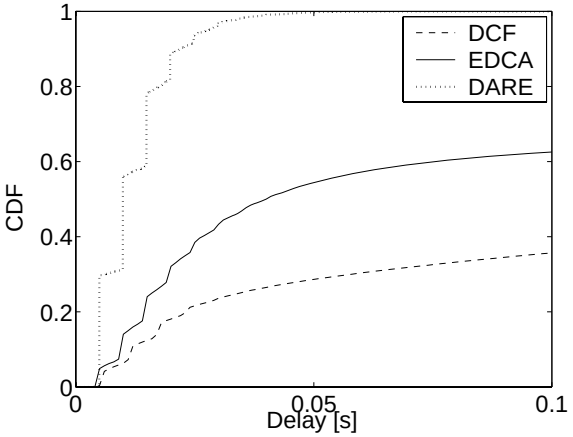
All wireless communication uses a fixed transmission power of $100\,\mathrm{mW}$; using NS-2's standard channel model, this results in a communication range of about $230\,\mathrm{m}$. As MAC protocols, we consider DCF as provided by NS-2 as well as our own implementation of EDCA [14] and DARE. As a routing protocol, we use NS-2's implementation of AODV.

To model the network traffic, a constant bit rate of $1\,\mathrm{Mbps}$ is assumed. We distinguish between real-time flows and non-real-time (background) flows. A real-time flow sends fixed-size packets every $100\,\mathrm{ms}$; in EDCA, real-time flows are given highest priority. A non-real-time flow uses exponentially distributed inter-packet times and a fixed packet size ($512\,\mathrm{bytes}$). The number of real-time flows is varied, as is the total amount of background traffic. For background traffic, a number of sources is chosen randomly, each with load 20 or $50\,\mathrm{kbps}$ that sums up to the given background load. For either flow type, the destination node is the closest access point.

To summarize, the varying *parameters* in this performance evaluation (and their default values) are as follows: (i) the number of real-time flows N (default: $N = 10$), (ii) the non-real-time traffic load (default: 0), (iii) the expected mean on/off periods of nodes μ (default: $\mu = 600s$), (iv) the packet size of real-time flows (default: $512\,\mathrm{bytes}$), and, obviously, (v) the MAC protocols.



(a) $N = 10$ real-time flows.



(b) $N = 20$ real-time flows.

Fig. 7. CDF of real-time packet delay.

As *metrics*, we use (i) the delay of packets from source to access point, (ii) the throughput for individual real-time flows, and (iii) the amount of slot shifts and blocked flows. Only successfully reserved real-time flows are considered in these metrics (except for blocked flows, of course). We do not consider the performance of background traffic.

To obtain *statistically meaningful results*, we ran 200 repetitions for each selected combination of parameters. Each repetition is run for $2000\,\mathrm{s}$ simulated time (except when varying μ, see below).

B. Delay

As a first performance metric, we look at the end-to-end delay of packets in a real-time flow.

Fig. 7(a) shows the Cumulative Distribution Function (CDF) of the delay, comparing DCF, EDCA, and DARE using default parameters. DARE manages to deliver *all* packets of reserved

flows to their destination within less than 0.05 s. DCF and EDCA, on the other hand, deliver a substantially smaller fraction of packets within this time; DCF needs up to 3 s to deliver a packet in this setup. Increasing the number of real-time flows (e.g., from $N = 10$ to 20), makes the differences between these protocols even more pronounced (Fig. 7(b)).

These results also give more insight into the delay characteristics of DARE. The step-like behavior of the delay's CDF is due to flows traversing different numbers of hops. The fact that the CDF is not a perfect step function (with one delay value for each number of hops between source and destination) is due to slot shifting taking place in the network, resulting in slightly different delay values for packets traveling along different paths of the same hop count. Nonetheless, all packets belonging to an established flow arrive with the same delay to the destination; the difference only exists between different flows, not between packets of the same flow. Overall, the delay is predictable when knowing the number of hops that a packet has to travel. DCF and EDCA, on the other hand, have a much more spread out CDF, reflecting the unpredictability of their random access delay.

Not only is DARE's packet delay more predictable, it is also substantially smaller. Fig. 8 compares average packet delays of the three MAC protocols as a function of the number of (established) real-time flows N. Averaged over different flows with different hop count, DARE achieves a practically constant delay irrespective of N; DCF and EDCA deteriorate at higher offered load.

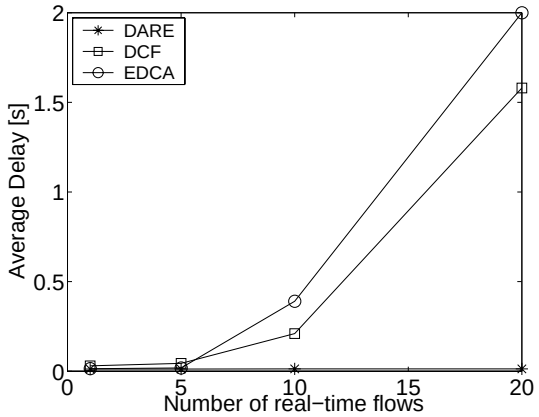


Fig. 8. Average delay over the number of real-time flows.

Comparing Figs. 7(a) and 7(b) also indicates that with an increased number of offered flows, the step characteristic of DARE's delay CDF is less pronounced. This observation corresponds to the percentage of flows that experience a slot shift (Fig. 9). Slot shifting becomes necessary more frequently if the network fills up with reservations, and new flows can only be admitted if they "squeeze in" between existing flows. This explains the somewhat increased (but still good) variability of DARE's delay at higher offered load. Still, all packets belonging to a shifted flow have the same delay with no variation.

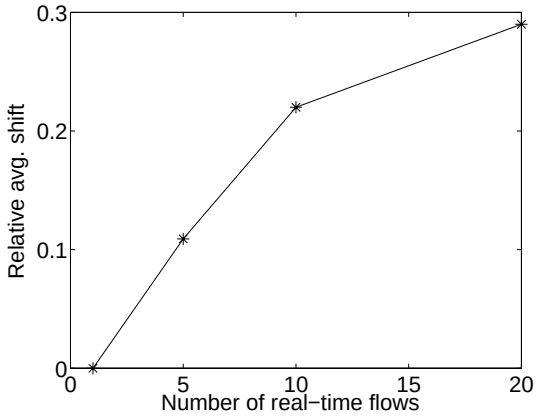


Fig. 9. Percentage of flows experiencing slot shifts.

C. Throughput and Blocking

As an outcome of another simulation with default parameters, Fig. 10 reports the average throughput for each protocol as a function of the number of attempted real-time flows. While DCF shows the lowest throughput, EDCA outperforms DARE if the number of attempted flows is low (here: 10). For more real-time flows (here: 20), DARE performs better than EDCA. This result requires closer inspection. With the chosen parameters (512 byte packets every 100 ms), a single real-time flow generates about 40.9 kbps load on the network. This shows that DARE can actually only support about 7 out of 10 offered flows (resulting in about 280 kbps offered load). Since a successfully reserved flow should be able to transport all its packets, this lack in throughput could be explained by rejected flow reservations.

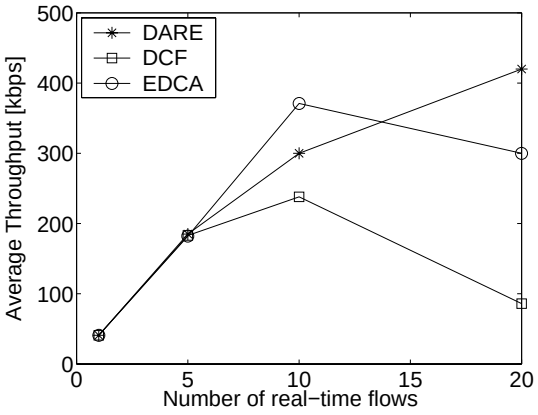


Fig. 10. Average throughput over the number of offered real-time flows.

This interpretation is corroborated by Fig. 11, showing the ratio of flows that are attempted but not accepted by DARE. Such an unsuccessful flow could be due to a rejected reservation request (blocking) or due to a path failure. These numbers explain the smaller throughput of DARE compared to EDCA — e.g., about 25 % unsuccessful flows pretty much explain this gap — and are in accordance with DARE's design philosophy only to admit a flow when it can be supported at high throughput and low delay. Accordingly, the number of

blocked flows also increases at higher offered load since the network is less likely to be able to support it.

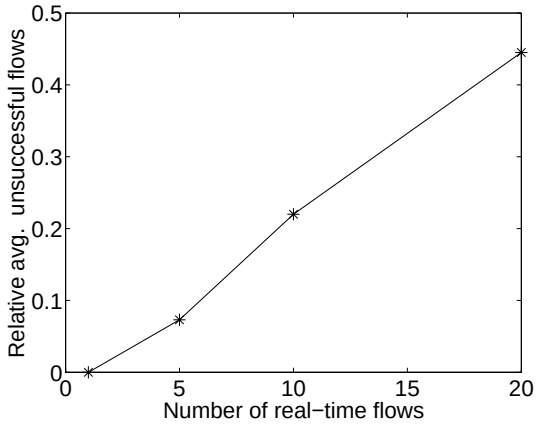


Fig. 11. Percentage of blocked flows as function of number of offered flows.

D. Impact of Background Traffic

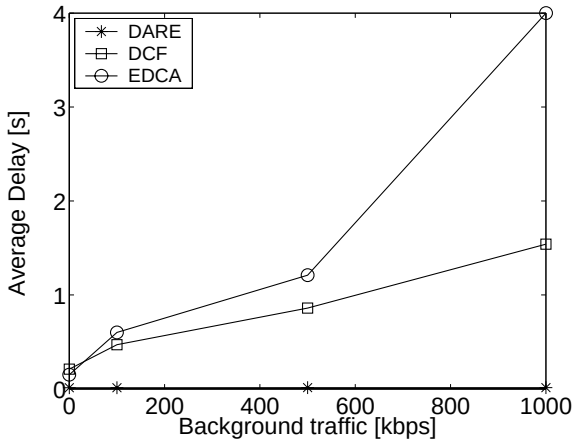
So far, we have varied the number of active real-time flows but did not include any non-real-time traffic in the scenarios. This subsection looks at the consequences of such background traffic. Again, we fix all parameters to their default values ($N = 10$, corresponding to about 400 kbps real-time load) except background load, which is varied between 0 and 1000 kbps total load generated by all sources. This corresponds to one third of the total available network capacity (four access points operating at 1 Mbps each can at maximum drain 4 Mbps from the mesh) and is sufficient to demonstrate crucial differences between different protocols. Larger values of the background traffic are analyzed in reference [7] for different scenarios with similar results.

Figure 12 shows the impact of the background traffic on the average delay and throughput of the real-time traffic. DARE's delay and throughput do not significantly vary with increased background traffic, confirming the initial design choices and showing that real-time traffic is protected by means of reservations from interference. DCF and EDCA, on the other hand, suffer considerably from increased background load. Even at modest background load, the performance of DCF or EDCA is unacceptable, e.g., for interactive applications.

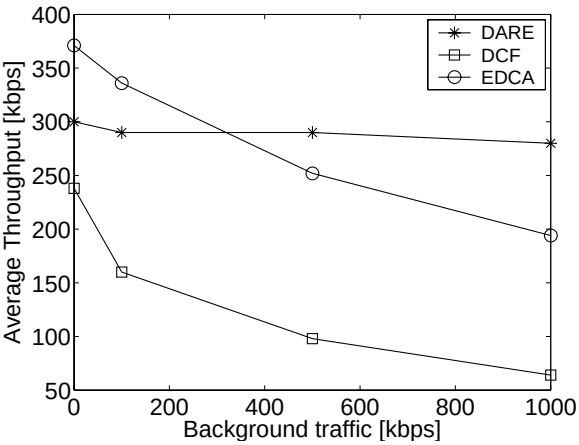
E. Impact of Node Outage

In the previous simulations, nodes were always active and did not switch off. Let us now investigate the impact of nodes switching on and off, in particular the impact of the average on/off period μ on delay and throughput. To improve statistical confidence, the simulated time is now 3600 s.

If we increase the average on and off time μ of individual nodes, nodes change their status less frequently, hence the overall topology becomes more stable. As a consequence, fewer link breaks occur in existing reservations, and fewer disturbances occur due to nodes powering on and possibly interfering with reservations. All in all, we expect longer



(a) Average real-time delay.



(b) Average real-time throughput.

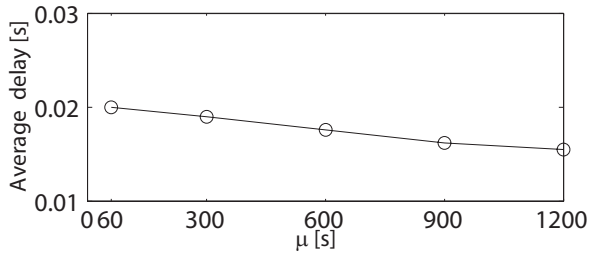
Fig. 12. Impact of background traffic load on real-time delay and throughput.

on/off times to be beneficial. This intuition is validated by the simulation results shown in Fig. 13. As expected, the delay shortens and the throughput increases as the topology becomes more stable. Nevertheless, DARE still works well even for small μ, where the topology changes frequently.

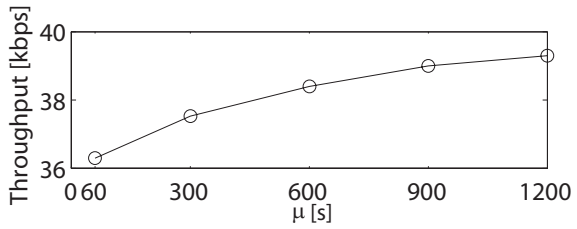
F. Impact of Number of Hops

The number of hops of a path has huge impact on the end-to-end delay. Naturally, increasing a path with another hop means that one more node must receive and transmit the packet. For EDCA and DCF, which have contention-based access, an increased hop number has bigger repercussions on delay and throughput than for DARE. We demonstrate this by a simulation of a network with only one real-time flow where the number of hops is varied. Two background traffic types, 100 kbps and 500 kbps, are transmitted from nodes all within the direct neighborhood of the reserved path.

Fig. 14 shows the impact of the number of hops on the delay and the throughput. The delay is shown as average delay per hop, where DARE has a constant per hop delay of approximately 0.005 s. EDCA and DCF have a large increase of the per-hop delay. The average path throughput decreases



(a) Average real-time delay.



(b) Average real-time throughput.

Fig. 13. Impact of node outage on real-time delay and throughput.

drastically for EDCA and DCF as the number of hops increase, especially when the network load is higher.

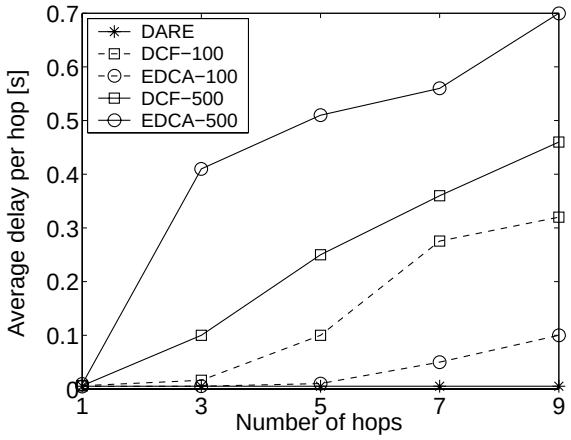
G. Impact of Packet Size

We investigate the impact of the packet size using a deterministic scenario; we check how many real-time flows can be reserved at one AP, without background traffic or path failure. The sources are located such that no intermediate node is involved in more than one real-time flow. The number of hops of a path is fixed for each simulation; all paths have either 2 or 3 hops. We fix the real-time traffic's period to 100 ms and look at packet sizes 144, 320, 512, and 1024 bytes. We let the sources of the real-time flows start at a random time and perform 400 simulations for each hop number and packet size and look at the average number of paths that can be accepted. The results are given in Table I.

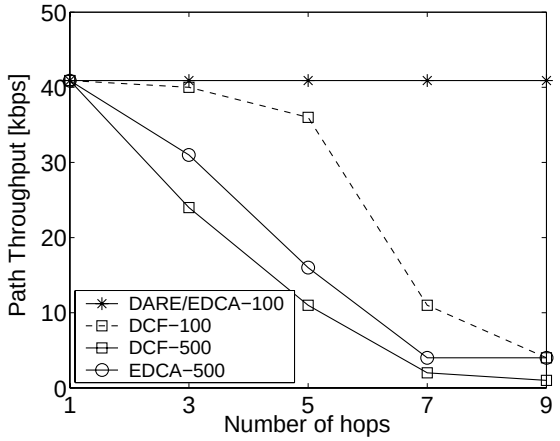
TABLE I
NUMBER OF ESTABLISHED RESERVATION PATHS

Packet size	Hops per path	
	2	3
144	8.4	6.2
320	7.7	5.9
512	6.4	4.7
1024	2.3	1.3

The maximum number of paths that could be supported for packet sizes 144 and 320 bytes is above 12 for both two and three hop paths. The maximum number of possible paths for 512 bytes is 8 for the three hop scenario. But due to inter-slot space between accepted reservations that could not be used, the average number of paths does not differ that much (except for 1024 bytes).



(a) Average real-time delay.



(b) Average real-time throughput..

Fig. 14. Impact of path length on real-time delay and throughput.

IV. RELATED WORK

Priority-based approaches for QoS support in DCF are described in references [5], [15]–[17]. Within a node, a packet is handled according to its priority level. Packets belonging to different priority classes are separated via different queues within a node and different waiting/random backoff times before they are transmitted.

A reservation mechanism allocates resources for transmission, i.e. time slots at each hop of a multihop path. To minimize the probability that a path might not be fully reserved, these reservations should be performed *end-to-end* before transmission of data begins. Existing mechanisms [18]–[22], however, tend to reserve each hop separately; most common here is the RTS/CTS handshake. An end-to-end reservation is still considered to be a challenge in reference [23]. Further, the reservation should be performed for the *whole flow*, which minimizes the signaling load and guarantees a non-varying quality during the whole application transmission.

End-to-end flow reservation mechanisms have been widely considered in wired networks, but the situation in wireless mesh networks is more challenging; nodes not part of a reserved path might interfere. Sufficient reservation infor-

mation must be spread to these nodes. To explicitly exchange reservation tables as in the MACA/PR protocol [24] or transmissions of energy bursts as in Blackburst [21] are some existing methods. Another solution, and our suggestion, is to use a piggy-back technique. This method minimizes signaling and information overload; nodes are only informed about reservations that they could directly interfere with [18]. Moreover, all reservation information should be spread in a two-hop neighborhood. A node that is too far away to be able to decode a packet may still be a possible interferer [25]. This is also true for nodes participating in a path; hence, they must abstain in reserved slots two hops back.

What is largely missing in existing reservation mechanisms is failure handling; reserved paths can break due to nodes leaving the chain and old reservations must be released. Our work addresses this basic challenge of an end-to-end, interference-protected, low-overhead, failure-handling protocol.

V. Conclusions

This article presented DARE — a distributed end-to-end reservation protocol for IEEE 802.11-based wireless mesh networks. The approach is to allocate and use periodic time slots for QoS-demanding applications. DARE reserves these time slots in a fully distributed way, schedules the real-time data packets, repairs broken reservations, and disseminates the reservation information to potential interferers using a piggy-back technique.

Our simulation-based study shows that DARE offers a reliable and efficient support for QoS applications. It provides a constant throughput as well as low and stable end-to-end delay for a reserved real-time flow. While the performance of DCF and EDCA strongly degrades with increasing network load, DARE offers stable throughput and delay even for high traffic loads. Only if the network load is low, EDCA yields a higher throughput.

Some extensions could further improve the DARE approach. One is to piggy-back data on the RTR messages or to send early data using DCF while the reservation is still being set up. This shortens the path setup delay but in turn introduces uncertainties in the early phase of a flow. A further open issue is to investigate the suitability of DARE in networks with mobile nodes.

References

[1] M. S. Gast, *802.11 Wireless Networks: The Definite Guide*, 2nd ed. O'Reilly, 2005, ch. A peek ahead at 802.11n: MIMO-OFDM.

[2] J.-C. Chen, M.-C. Jiang, and Y. Liu, "Wireless LAN security and IEEE 802.11i," *IEEE Wireless Communications*, Feb. 2005.

[3] "IEEE Std 802.11: Wireless LAN Medium Access Control (MAC) and Physical layer (PHY) specifications," 1997.

[4] K. Sundaresan, H.-Y. Hsieh, and R. Sivakumar, "IEEE 802.11 over multi-hop wireless networks: Problems and new perspectives," *Elsevier Ad Hoc Networks*, Apr. 2004.

[5] "IEEE Std 802.11e – Specific Requirements Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications amendment 8: MAC Quality of Service (QoS)," 2005.

[6] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) routing," RFC 3561, July 2003.

[7] 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, Feb. 2004.

[8] 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 Veh. Techn. Conf. (VTC)*, Los Angeles, CA, Sept. 2004.

[9] ——, "Distributed MAC for real-time traffic in multi-hop wireless networks," in *Proc. IEEE Conf. Sensor Ad Hoc Commun. Netw. (SECON)*, Santa Clara, CA, Oct. 2004.

[10] E. Carlson, C. Bettstetter, C. Prehofer, and A. Wolisz, "A performance comparison of QoS approaches for ad hoc networks: 802.11e versus distributed resource allocation," in *Proc. European Wireless*, Nicosia, Cyprus, Apr. 2005.

[11] V. Anantharaman, S.-J. Park, K. Sundaresan, and R. Sivakumar, "TCP performance over mobile ad-hoc networks: A quantative study," *Wireless Communications and Mobile Computing*, Mar. 2004.

[12] L. M. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," in *Proc. IEEE Infocom*, Anchorage, AK, Apr. 2001.

[13] "Network simulator 2," http://www.isi.edu/nsnam/ns/.

[14] S. Wiethoelter and C. Hoene, "Design and verification of an IEEE 802.11e EDCF simulation model in ns-2.26," Telecommunication Networks Group, Technische Universität Berlin, Tech. Rep., Nov. 2003.

[15] N. H. Vaidya, P. Bahl, and S. Gupta, "Distributed fair scheduling in wireless LAN," in *Proc. ACM MobiCom*, Boston, MA, Aug. 2000.

[16] Q. Qiang, L. Jacob, R. R. Pillai, and B. Prabhakaran, "MAC protocol enhancements for QoS guarantee and fairness over the IEEE 802.11 wireless LAN," in *Proc. Intl. Conf. Comp. Commun. Netw. (ICCNC)*, Miami, FL, Oct. 2002.

[17] M. A. Visser and M. E. Zarki, "Voice and data transmission over an 802.11 wireless network," in *Proc. IEEE PIMRC*, Toronto, Canada, Sept. 1995.

[18] 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*, Bejing, China, Sept. 2003.

[19] A. Acharya, A. Misra, and S. Bansal, "Design and analysis of a cooperative medium access scheme for wireless mesh networks," in *Proc. Intl. Conf. Broadband Netw. (BroadNets)*, San Jose, CA, Oct. 2004.

[20] 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*, Amsterdam, Netherlands, June 2001.

[21] J. L. Sobrinho and A. S. Krishnakumar, "Quality of Service in ad hoc carrier sense multiple access wireless networks," *IEEE J. Select. Areas Commun.*, vol. 17, no. 8, pp. 1353–1368, Aug. 1999.

[22] S. B. Lee and A. T. Campbell, "INSIGNIA: In-band signaling support for QoS in mobile ad hoc networks," in *Proc. Intl. Workshop Mobile Mulimedia Communication (MoMuC)*, Berlin, Germany, Oct. 1998.

[23] R. Ramanathan, "A radically new architecture for next generation mobile ad hoc networks," in *Proc. ACM MobiCom*, Cologne, Germany, Sept. 2005.

[24] C. H. R. Lin and M. Gerla, "Asynchronous multimedia multihop wireless networks," in *Proc. IEEE Infocom*, Kobe, Japan, Apr. 1997.

[25] 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.

Appendix A: DCF and EDCA Basics

This appendix explains the random access control in IEEE 802.11 networks, following the Distributed Coordination Function (DCF) and the Enhanced Distributed Channel Access (EDCA) protocol.

Before a node is allowed to transmit, it must sense the shared channel. If the channel is idle, the node waits a certain time period, namely a Distributed Inter Frame Space (DIFS) in case of DCF, or an Arbitrary Inter Frame Space (AIFS) in case of EDCA. During this waiting period, it continues to sense the channel. If the channel is still idle after the waiting period, the node transmits.

If the channel becomes busy (not idle) during the waiting period, the node performs a backoff procedure. It has to wait ("back off") a certain random time. This random time is determined as follows. The node sets a backoff counter to a

random integer number from the interval $[0, \mathrm{CW}]$. This interval is called the contention window (CW). Whenever the channel is idle for a period of *aSlotTime*, the node decreases its backoff counter by one. If the channel is busy during that period, it freezes the counter until the channel is idle again. Once the counter reaches zero, the node transmits.

Optionally, before a node transmits its payload data, it performs a virtual carrier sensing using a two-way handshake to the intended receiver. The node sends a Request-to-Send (RTS) message to the intended receiver. If the latter is not engaged in another transmission, it answers with a Clear-to-Send (CTS) message back to the sender. Both control messages contain the duration of the planned transmission and inform surrounding nodes that they have to abstain from sending during this time.

While DCF treats each traffic flow in the same manner by using the same parameters CW and DIFS for each flow, the EDCA introduces different priority classes ("access categories"). Four access categories have been defined (lowest priority first): background, best effort, video and voice. In each node, each access category has its own queue and packets of higher priority queues are handled first. Moreover, an access category of high priority uses a small CW and a small AIFS such that it is likely that traffic belonging to this category can access the channel first. The AIFS for the priority classes are determined according to:

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

where AIFSN is a real number and *aSIFSTime* is the physical layer parameter for the Short Inter Frame Space (SIFS) used between RTS and CTS, and, Data and ACK. For the smallest possible AIFS, we have AIFSN$= 2$ [3], [5]. Table V summarizes the minimum and maximum CW values as well as the AIFSN of the four EDCA categories.

TABLE II
BACKOFF AND AIFSN VALUES FOR EDCA AND DCF

Category	Voice	Video	Best effort	Background	DCF
CWmin	7	15	31	31	31
CWmax	15	31	1023	1023	1023
AIFSN	2	2	3	7	2

Emma Carlson is a PhD student at the Berlin University of Technology, where she focuses on QoS in distributed wireless networks. She studied electrical engineering at the Royal Institute of Technology (KTH) in Stockholm, where she received her Civ. Ing. degree in 2000. After working at Siemens Mobile Networks division for two years, she started her PhD studies at the TU Berlin, Telecommunication Networks Group, in 2003.

Christian Prehofer is a principal scientist at Nokia Research in Helsinki. His research interests include self-organized and ubiquitous systems, multihop networking, as well as software architecture and software technologies for mobile communication systems. Before, he was senior manager at DoCoMo Euro-Labs in Munich. From 1998 to 2001, he held positions as system engineer and software architect in the area of communication systems and mobile devices. He is also a lecturer at the Technical University of Munich, where he obtained his Ph.D. and habilitation in computer science in 1995 and 2000, respectively. He studied computer science at the TU München and Univ. of Illinois, where he graduated with a masters degree in 1992. He is author of more than 70 publications and has contributed to several standardization bodies.

Christian Bettstetter is a full professor at the University of Klagenfurt, Austria, where he leads a research group on mobile and wireless systems. His interests include algorithms and protocols, networking concepts, and mathematical methods for ubiquitous communication. He studied electrical engineering and information technology at the Technische Universität München (TUM), Germany, where he received the Dipl.-Ing. degree in 1998 and the Dr.-Ing degree (summa cum laude) in 2004. Prior to becoming a professor, he worked as a senior researcher at the European lab of NTT DoCoMo. Christian co-authored about 60 technical papers and the Wiley book *GSM: Switching, services and protocols*. His article on pitfalls in the random waypoint mobility model received the 2004 outstanding paper award from the German ITG.

Holger Karl is a full professor of computer science at the University of Paderborn, Germany. He is head of the research group Computer Networks, which concentrates on mobile and wireless communication. His main research interests are architectural questions for future mobile communication systems, cross-layer optimization, and wireless sensor networks. He studied computer science in Karlsruhe, Germany and at the University of Massachusetts, Amherst, MA; he undertook his graduate studies at Humboldt-University of Berlin. He obtained his Diploma and PhD degrees in 1996 and 1999, respectively. Afterwords, he was assistant professor at Technical University Berlin. He is co-author of the Wiley book *Protocols and Architectures for Wireless Sensor Networks.*

Adam Wolisz (Diploma in engineering, 1972, Doctoral Degree, 1976, Habilitation 1983 - Silesian University of Technology, Gliwice) has been working since 1980 on computer networks and distributed systems. He was at the Polish Academy of Sciences until 1990, and later with the Research Institute GMD-Fokus in Berlin from 1990 to 1993. Since 1993 he has been at the Technical University Berlin where he helds a chaired Professor for Telecommunication Networks. Since the summer of 2003, he has also been Executive Director of the Institute for Telecommunication Systems. He served as the Dean of the Faculty of Electrical Engineering and Computer Science in the period 2001-2003. Since summer 2005 he has also been Adjunct Professor at the Dept. EE&CS, University of California, Berkeley. His research interests are in architectures and protocols of communication networks. Recently he has been focusing on wireless/mobile networking and sensor networks.