Real-Time Multiplayer Game Support Using QoS Mechanisms in Mobile Ad Hoc Networks

Dirk Budke, Károly Farkas, Bernhard Plattner
Computer Engineering and Networks Laboratory (TIK)
Swiss Federal Institute of Technology Zurich (ETH Zurich)
Gloriastrasse 35, CH-8092 Zurich, Switzerland
Email: {budke, farkas, plattner}@tik.ee.ethz.ch

Oliver Wellnitz, Lars Wolf
Institute of Operating Systems and Computer Networks (IBR)
Technische Universität Braunschweig
Mühlenpfordtstr. 23, D-38106 Braunschweig, Germany
Email: {wellnitz, wolf}@ibr.cs.tu-bs.de

***Abstract*— Real-time applications, especially real-time multiplayer games, are getting popular in mobile ad hoc environments as mobile devices and wireless communication technologies are becoming ubiquitous. However, these applications have strict demands on the underlying network requiring low latency with a minimum of jitter and a small packet loss rate. Therefore, Quality of Service (QoS) support from the network is essential to meet the demands of real-time applications which is a challenging task in mobile ad hoc networks due to the high level of node mobility, properties of the wireless communication channel and the lack of central co-ordination.**

In this paper, we analyze and evaluate, via simulations, common QoS extensions, like backup route or priority queuing, and propose some new ones, such as real-time neighbor aware rate control policies or hop-constrained queuing timeouts, to ad hoc routing in IEEE 802.11 mobile ad hoc networks. We show how the performance, using these QoS extensions, of the selected AODV routing protocol improves significantly for connections up to three hops making possible to meet the demands of real-time applications. Moreover, we present how some common mechanisms like local repair of AODV and RTS/CTS degrade the performance and should not be used in case of real-time traffic.

I. Introduction

Online real-time applications such as multiplayer games have become very popular recently and are a big business in today's Internet expecting increasing revenue. Real-time applications share the common property that the information, for instance, the game character movements have to be delivered as fast as possible to the counterparts. If this data is delayed and arrives too late the game is not playable any more. Thus, real-time applications such as multiplayer computer games have strict demands on the underlying network and require low latency[1] connections, generally in the range of up to 50 – 150 ms, with a minimum of jitter and low packet loss rate below 5 % [1]. While multiplayer games are playable in the Internet, in wireless networks Quality of Service (QoS) mechanisms are required to meet the demands of real-time applications.

With the constantly increasing amount of powerful mobile devices, such as PDAs, laptops and mobile phones with wireless networking support, real-time applications and even multiplayer computer gaming in wireless environments are gaining much interest. Even though wireless networking infrastructure, such as base stations or access points, might not be available, various heterogeneous mobile devices can connect to each other setting up spontaneously a self-organized mobile wireless multihop ad hoc network (MANET) that does not rely on any existing infrastructure. However, ad hoc communication introduces several challenges mainly due to the mobility of the nodes, limited device resources, properties of the wireless channel and the lack of central co-ordination. The routing protocol, which is needed to send packets from a source to a destination via multiple hops, must cope with these challenges. Moreover, it has to be able to satisfy the QoS requirements of real-time applications, as well. But QoS provisioning in MANETs is even more difficult than in wired networks because of, among others, arbitrary mobility, weak wireless links, signal fading and interference, and the used channel access mechanisms. Thus, meeting the applications' demands in MANETs is very challenging and has been an open field of research in the last couple of years.

In this paper, we analyze and evaluate common QoS extensions, such as priority queuing, backup route, broken link detection, etc., and propose some new ones, like real-time neighbor aware rate control and hop-constrained queuing timeouts, to ad hoc routing in IEEE 802.11 [2] MANETs focusing on the requirements of real-time applications, specifically real-time multiplayer computer games. We selected the AODV [3] routing protocol as the starting point of our work because this protocol had shown the best performance in our initial simulations compared to some other ad hoc routing protocols, such as DSR, DSDV and OLSR [3]. However, our simulations show that end-to-end communication delay and jitter experienced using AODV are still too high to meet the demands of multiplayer computer games. In order to improve the performance of AODV for real-time applications, we require improved congestion handling mechanisms which can cope with mobility and the properties of an unstable wireless channel. Thus, in our work, following a cross-layer design, we have used: (1) QoS extensions to AODV like local repair and backup route; (2) traffic management mechanisms like priority queuing, timeouts, real-time neighbor aware rate control; and (3) MAC (Medium Access Control) layer support mechanisms like broken link detection, signal strength monitoring, neighbor

[1] We are using the terms *latency* and *delay* exchangeable in this paper

detection, RTS/CTS (Ready To Send/Clear To Send) adaptation.

We show in this paper, how the different QoS extensions do or do not improve the performance of real-time traffic in MANETs. First, we categorize and briefly discuss the extensions then present their combined impact on AODV's performance via simulations using the NS-2 network simulator [4]. Our initial routing protocol comparison results show that AODV outperforms the other routing protocols but still does not meet the demands of real-time applications. By applying priority queuing with timeouts and real-time neighbor aware rate control combined with broken link detection and backup routes, the end-to-end communication delay, delay jitter and loss rate can be reduced significantly. For connections up to three hops, AODV with the applied QoS mechanisms meets the demands of real-time multiplayer games. However, some common mechanisms like local repair of AODV and RTS/CTS degrade the performance of real-time traffic and should not be used.

The rest of the paper is organized as follows: In Section II, we discuss the issues of QoS provisioning for real-time multiplayer games. Then, we introduce the different QoS extensions in Section III and discuss them in Section IV. In Section V, we evaluate these extensions via simulations. We survey related approaches in the field of QoS provisioning in Section VI and conclude the paper in Section VII.

II. QoS Provisioning for Real-Time Multiplayer Games

Real-time applications, especially multiplayer games such as first person shooters, real-time strategy games, or sports games have strict QoS demands on the underlying network. Moreover, in MANETs, compared to wired networks like the Internet, these applications encounter additional problems due to the special challenges of MANETs. In this section, we discuss the substantial QoS requirements of real-time multiplayer games then describe our objective regarding to QoS provisioning in the light of the challenges of MANETs.

A. Requirements of Real-Time Multiplayer Games

In case of real-time multiplayer games, end-to-end communication delay, jitter and packet loss to a certain degree are the most important QoS attributes of the network, while the available network bandwidth is of less importance [5], [6]. For most real-time multiplayer games, a maximum of 150 ms round trip delay is still acceptable [1]. The effects of jitter, however, are not so well-researched yet. In [7], the authors suggest that latency and jitter are strongly coupled in the Internet with the ratio of jitter to latency being 0.2 or smaller. This also means that jitter in general is very small in the Internet if latency is below 150 ms and therefore has little impact on the game. Furthermore, players' perception of jitter is game-dependant [8] because high level of jitter usually leads to packets not arriving in time thus requiring the use of prediction mechanisms such as dead-reckoning. As the quality of these prediction mechanisms differ from game to game, players also perceive jitter differently. Furthermore, prediction mechanisms cannot always anticipate players' actions accurately so high level of jitter usually degrades the players' experience. Hence, the jitter level must be kept as low as possible. Moreover, the packet loss rate has similar effects and it should be kept in the range of 3 – 5 % for real-time multiplayer games [1].

B. Objective and Challenges

Our objective is to extend mobile ad hoc routing with QoS mechanisms by which the performance of the ad hoc routing protocol and the network can meet the demands of real-time multiplayer games. However, to give QoS guarantees in MANETs is a difficult task and the QoS extensions have to face the following challenges:

- **Mobility** - The topology of the ad hoc network might change unpredictably resulting in broken links and stale routes.
- **Congestion** - Real-time traffic must arrive in-time even if the network is highly loaded.
- **Shared Medium** - Wireless networks operating in ad hoc mode do not provide any QoS guarantees at the MAC layer due to the applied contention based medium access mechanism.
- **Wireless Signal** - The wireless signal suffers from fading and interference which gives rise to gray zones [9] and frequent retransmissions.

III. QoS Extensions to Mobile Ad Hoc Routing

In this section, we collect common and propose some new QoS extensions which are supposed to improve the performance of mobile ad hoc routing. Moreover, we classify these extensions into three different categories and point out the challenges they are supposed to tackle.

A. QoS Extensions

As QoS provisioning is a complex task and should be handled on several layers in the protocol stack we followed a cross-layer design. Thus, we classify the collected and proposed QoS extensions providing QoS support on the routing, interface queue and MAC layer, as illustrated in Fig. 1, into the following categories: (1) enhancements to the routing protocol; (2) traffic management; and (3) MAC layer support.

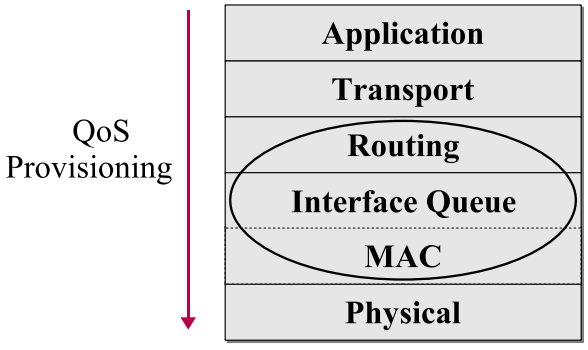

Fig. 1. Quality of Service Support in a Mobile Ad Hoc Network Node's Protocol Stack

TABLE I
QoS EXTENSIONS

		Challenges			
Category	**QoS Mechanism**	**Mobility**	**Congestion**	**Shared Medium**	**Signal**
AODV Enhancements	Local Repair	√			
	Backup Route	√			
Traffic Management	Priority Queuing		√		
	Timeouts		√		
	Rate Control		√	√	
Mac Layer Support	Broken Link Detection	√	√		
	Neighbor Detection	√	√		
	Signal Strength Monitoring				√
	RTS/CTS Adaptation			√	

Moreover, the different QoS extensions address different challenges of the mobile ad hoc environment. In Table I, we depicted the extensions and the corresponding challenges they are supposed to tackle.

1) AODV Enhancements: As the results of the ad hoc routing protocol comparison in Section V show, the reactive protocol *Ad Hoc On-Demand Distance-Vector* (AODV) [3] provides the best overall performance among the considered protocols. So, we selected the AODV-UU [10] implementation as the starting point of our work. To cope with mobility and broken links, AODV makes use of a *local repair* mechanism that allows intermediate nodes that have detected a link failure to temporally queue packets and repair the route. In addition, we extended AODV with the use of *backup route*s to repair broken links transparently and without any delays.

2) Traffic Management: When employing *priority queuing*, real-time packets are preferably transmitted to make sure that even in situations with higher load real-time packets do not become obsolete. Still, it might happen that real-time packets have to wait too long in the high priority queue. To prevent for the transmission of outdated packets we introduce hop-constrained queue *timeouts*. These are used to drop obsolete real-time packets and thus save bandwidth by applying a gradually decreased timeout timer. The actual timeout value at a given node is computed taking into account the number of already traversed nodes by the packet. In addition, to limit the amount of low priority traffic we introduce real-time neighbor aware *rate control* policies which prevent the occupation of the communication channel by nodes sending low priority traffic if other nodes have high priority traffic to be sent.

3) MAC Layer Support: Many features, already implemented in the IEEE 802.11 MAC layer, remain unused in higher layers and are implemented there again, however, less efficiently. With *broken link detection* based on Link Layer Feedback (LLF) the routing protocol is notified instantly if packets cannot be sent any longer over a certain link. In general, routing protocols periodically broadcast messages for *neighbor detection*. However, IEEE 802.11 ad hoc networks already broadcast periodic advertisements and thus the periodic routing messages are not required any more, neither for broken link detection nor for neighbor detection. With the help of *signal strength monitoring* more stable routes can be selected. In addition, IEEE 802.11 relies on the RTS/CTS (Ready To Send/Clear To Send) mechanism to avoid the hidden terminal problem. However, if used, *RTS/CTS adaptation* to the application's requirements is essential to not degrade the overall performance.

IV. DESIGN CONCEPTS

In this section, we describe our design concepts with regard to the mentioned QoS extensions and discuss these extensions in detail. We used NS-2 to implement these QoS extensions, so implementation details will be given where relevant. We did not implement and investigate all of these proposals in the simulator because, due to its limitations, we decided to do an implementation for a real hardware driver. Unfortunately, we could not investigate the real environment experiments due to time constraints.

A. AODV Enhancements

In order to extend the capabilities of the routing protocol to react efficiently to network topology changes and route failures we used *local repair* and enhanced AODV with *backup route*s.

1) Local Repair: AODV makes use of a local repair mechanism that allows intermediate nodes detecting link failures to queue packets temporally while it tries to repair the route. We used the built-in version of local repair from AODV.

2) Backup Route: AODV just stores the next hop entry to a certain destination in its routing table. If the link to that node breaks a new route discovery process has to be started which requires too much time for real-time data. The idea is to provide a backup route from the beginning that can be used instead. A backup route is a path with the same hop count as the default path but with a different next hop. For example, assume a scenario as illustrated in Fig. 2. Node S broadcasts RREQ messages to find a route to node D. Node I receives two RREQs, one from node 2 and the other from node 4. Both describe a 3-hop path to node S with the same sequence number. In this event, node I employs either node 2 or 4 as the next hop and uses the other as backup route. If a link to a neighbor on an active route breaks the backup route becomes the new active path. If hops with a higher hop count were used as backup paths, routing loops might occur (e.g., node I receives a RREQ from node 5 with a 5-hop path to node S

but this path contains a routing-loop and cannot be employed as backup route).

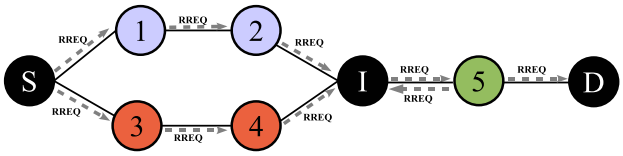

Fig. 2. AODV Extension Using Backup Routes

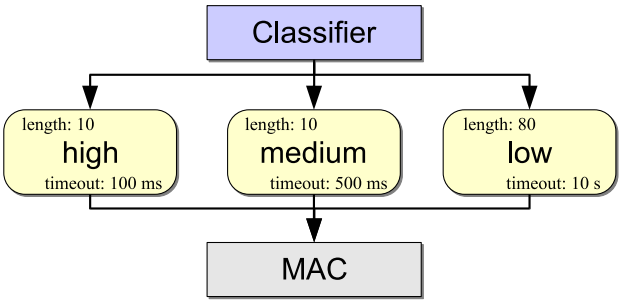

Fig. 3. Design of the Priority Queue

B. Traffic Management

The goal of traffic management is to differentiate among various types of traffic and give a higher level of service to high priority traffic at the cost of lower priority traffic. To achieve this, *priority queuing* can be used which we extended with hop-constrained queue *timeouts* and real-time neighbor aware *rate control* policies.

1) Priority Queueing: We used and modified the interface queue sublayer of NS-2 to implement priority queuing mechanisms. In our implementation, the interface queue consists of three sub-queues with different priorities and every packet is classified into one of three categories, as shown in Fig. 3, employing the TOS/DS field[2] in the IP header. All data packets that do not have any particular real-time QoS demands are marked as *low priority*, for instance, file transfer and e-mail traffic. AODV routing management packets such as RREQ, G-RREP[3] and RERR packets are marked as *medium priority*. And finally, real-time data packets are marked as *high priority* as well as AODV RREP messages if the high priority flag had been set in the corresponding RREQ. In contrast to medium and low priority, high priority packets are quickly outdated and dropped if they cannot be delivered in-time. Low priority packets might on the one hand experience higher delays, but on the other hand they are forwarded and delivered to the destination more reliably. Every sub-queue has a limited size and allows a packet only to be queued for a certain time interval. All sub-queues are exhaustive, meaning that a queue with lower priority will only be processed if all queues with higher priority are empty. In order to limit the amount of real-time and routing packets a node is allowed to transmit, the size of the high and medium priority sub-queues has been restricted to 10 slots. A much longer queue for real-time packets would result in more outdated packets. As the packets are marked with different priorities they can be handled specifically by the routing protocol and the interface queue.

The route request packets RREQ are marked with medium priority by default. By doing this, the actual current queue length of intermediate nodes are indirectly taken into account during the route discovery process in AODV. Nodes that have fewer packets in their high and medium priority sub-queues will be able to forward their packets quicker than nodes that already have to forward multiple high priority data streams. However, to minimize the delay in the event of broken links or the unavailability of routes for high priority data, we introduced an additional flag (bit 6 in the AODV RREQ header) that indicates that a RREQ is of high priority to enable nodes speeding up the route discovery process. But without marking the RREQ the network layer of the node that replies to the RREQ is not aware that the reply is urgent. Therefore, when the destination node receives a RREQ with a high priority flag, the TOS/DS field of the corresponding IP packet that holds the RREP message is set to high priority to make sure that the RREP is queued in the high priority sub-queue and forwarded as quickly as possible.

In addition to the interface queue, AODV itself queues data packets if no route to the destination is currently available. Because the route discovery process generally requires much more than 150 ms, we suggest that high priority packets are not to be queued by AODV but should be dropped immediately.

In order to cope with selfish nodes that simply mark all packets with high priority, packets could be filtered at the incoming interface to limit the amount of real-time packets from a certain node. However, we currently do not deal with malicious nodes in our implementation.

2) Timeouts: Real-time multiplayer games demand low latency connections with less than 150 ms round trip or 75 ms one-way delay, assuming symmetric latencies. However, packets that are delayed slightly more than 75 ms, might still be useful to prediction mechanisms such as dead-reckoning. Therefore, we decide to only drop packets which take more than 100 ms one-way delivery time and are clearly obsolete for real-time applications. Outdated real-time packets needlessly consume bandwidth and delay other real-time packets. Therefore, every packet that is stored in the queue is marked with a local time-stamp. When the packet is dequeued the time the packet spent in the queue is compared with the queue policy. The timeout interval for a high priority packet is 10 ms for every hop with a maximum value of 100 ms. Hence, the timeout mechanism supports only up to 10 hops from sender to destination which is fairly large for an ad hoc network. With every hop the packet passes, the timeout value is further decreased by 10 ms which approximates the

[2]Initially, Type Of Service now redefined as the Differentiated Services field

[3]In order to establish a bidirectional route between the source and destination nodes, an intermediate node that replies to an RREQ carrying the gratuitous flag sends an additional route reply message G-RREP to the destination

time it takes to process and forward the packet. The timeout information can be computed locally at each node by using the node's routing table because AODV, as a distance-vector routing protocol, knows about the number of hops to the source and the destination. The timeout mechanism is designed as a rather conservative approach and aims at dropping just the packets that will definitely come too late.

The medium priority sub-queue has a timeout of 500 ms which has shown reasonable performance in our simulations. Since both priority sub-queues are rather small and the packets are outdated quickly compared to the low priority sub-queue, additional mechanisms that prevent for starvation of low priority traffic have not been added.

3) Real-Time Neighbor Aware Rate Control: The current IEEE 802.11 protocol standards do not support QoS at the MAC layer and the medium access is carried out by the Distributed Coordination Function (DCF) handling every node equally [2]. All nodes have the same probability to gain access to the wireless channel and DCF does not distinguish between high priority and low priority data traffic. Thus, there is no guarantee that a node will send even high priority data packets within a certain time frame. Therefore, nodes that send high amounts of low priority data might consume most of the shared bandwidth. Due to the contention mechanism in DCF, other nodes which send high priority data might wait too long to access the channel and cannot transmit their high-priority data in time, although priority queuing has been used. The reason behind this is, that priority queuing works only locally and does not take neighboring nodes into account. To solve this problem at the MAC layer is one of the aims of the new QoS IEEE 802.11e [2] protocol that is still under development and not yet available. Even with the advent of IEEE 802.11e, the current standard will not be displaced immediately and thus, a mechanism is required that enables nodes that send real-time data to refrain other nodes from accessing the channel.

Our proposal is to limit the amount of low priority packets a node is allowed to transmit within a time interval depending on the amount of nodes handling real-time traffic in its neighborhood. Therefore, the interface queue needs access to the actual number of neighbors sending high priority traffic to adapt the rate control system of low priority traffic.

For this purpose, every node maintains a time-stamp *PRIO_TIMEOUT* that is updated when the node transmits a high-priority packet. If the node has sent real-time traffic within the last *PRIO_TIMEOUT* seconds the node marks broadcast routing messages with a real-time flag. Employing the unused bit 5 in the AODV header of RREQ and RREP messages, a node indicates whether it currently transmits high priority traffic or not. In addition to that, the routing table entry has been extended by a time-stamp *LAST_PRIO*. Every time a node receives high priority data or a routing message with the high priority flag, the *LAST_PRIO* time-stamp for that particular routing table entry is updated. Now, the actual amount of neighbors sending high priority traffic with an up-to-date *LAST_PRIO* time-stamp can be derived from the neighbor list and the interface queue can adapt the rate control of low priority traffic upon the number of neighbors sending real-time traffic. By introducing artificial delays for low priority traffic, nodes with high priority traffic have a better chance of sending their data in time. If the mobility rate of the nodes is low and nodes with real-time traffic have not sent routing messages for a while, the actual number of neighbors with real-time traffic might be higher, however. In addition, selfish nodes might just mark all routing packets with the priority bit. But we assume normal and our policy conform behavior of the nodes.

C. MAC Layer Support

In order to have accurate access to the current network state, such as an up-to-date neighbor list and the wireless signal strength for each neighbor, support from the IEEE 802.11 MAC layer is required. We use *broken link detection*, *neighbor detection*, *signal strength monitoring* and *RTS/CTS adaptation* as supporting mechanisms from the MAC layer.

1) Broken Link Detection: For broken link detection we employed the Link Layer Feedback (LLF) mechanism from the AODV-UU implementation. If a packet cannot be transmitted successfully, that is the sender does not receive an ACK from the destination within a certain time interval, the packet is retransmitted. Excessive retransmissions, however, can be reported by the MAC layer using LLF and this information can be propagated to the routing protocol to deal with the broken link. This mechanism has two benefits. First, it reacts to broken links by usually one order of magnitude quicker than relying on HELLO messages. And second, HELLO messages used for broken link detection are not required any more which reduces the routing overhead.

2) Neighbor Detection: To make routing decisions or management actions in the ad hoc environment an up-to-date neighbor list of a node is required. However, to maintain this list usually each application implements its own neighbor discovery procedure using network probes or HELLO messages. This results in message overhead and a waste of the scarce bandwidth. Since this service is commonly demanded it makes sense to provide a common interface to upper layers accessing the neighbor list from the MAC layer. Unfortunately, NS-2 does not provide a way to access the neighbor list on the MAC layer, thus we did not investigate its effect but decided to implement this feature in a Linux hardware driver. In a real environment implementation we can rely on BEACON messages that are sent out regularly every 100 ms by the IEEE 802.11 MAC layer if operating in ad hoc mode.

3) Signal Strength Monitoring: To provide for link stability and avoid the gray zones mentioned in [9] the signal-to-noise ratio (SNR) is measured for routing messages. If the signal strength of a RREQ is not beyond a certain SNR threshold the request is not processed. Thus, other neighbors that have received the RREQ with a higher signal strength will forward the packet and create a more stable route. AODV employs an *expanding ring search* based on the TTL (Time To Live) field in the IP header for dissemination of RREQ messages. If a node receives a RREQ which has been sent with the maximal TTL,

it does not drop the RREQ even though the signal strength is below the threshold, since no better route is available. Due to the limitations of the employed propagation model in NS-2, we did not implement signal strength monitoring for NS-2 but did a real environment implementation.

4) RTS/CTS Adaptation: NS-2 supports a certain packet size threshold to indicate if RTS/CTS should be used. By default the threshold is 0 and RTS/CTS is always enabled. We disabled RTS/CTS for real-time packets. Without RTS/CTS the transmission channel is shared equally among the contenting nodes. Moreover, it requires less bandwidth and we assume that it reduces end-to-end delay and delay jitter.

V. EVALUATION

In this section, we evaluate the introduced QoS extensions via simulations and discuss the results in the light of the demands of real-time multiplayer games.

A. Simulation Settings

In our simulations, we have used NS-2 including wireless extensions developed at CMU [11]. Throughout the simulations, each mobile node shares a 2 Mbit/s radio channel with its neighboring nodes using the two-ray ground reflection propagation model [12] and the IEEE 802.11 MAC protocol. The simulation scenario consists of 20 nodes in an area of 650x650 m^2. The transmission range of each node is 250 m, which is a typical value for WLAN in a free area without any obstacles. The nodes that do not take part in the multiplayer game follow the Random Way Point (RWP) model [11] with a uniformly distributed speed between 0-3 m/s and a pause time of 180 seconds. Player nodes are assumed to move only slightly, within a distance of 0-15 meters, since it is rather difficult to move and play at the same time with today's available mobile gaming devices. We simulated typical multiplayer game traffic [5] between the player nodes as high priority bidirectional UDP data flows with a Constant Bit Rate (CBR) traffic of 20 packet/sec and a packet size of 64 bytes. Additionally, we simulated five bidirectional low priority data flows between any two random nodes as background traffic using the same traffic pattern as the game traffic. We simulated 10 game sessions with a duration of 600 seconds using different seed values and then averaged the results. The detailed simulation parameters are depicted in Table II.

TABLE II
PARAMETERS OF THE USED SIMULATION SCENARIO

Parameter	Value
Simulation Time	600 s
Nodes	20
Mobility Model	Adapted Random Way Point (RWP)
Area	650x650 m^2
Speed	0-3 m/s
Pause Time	180 s
Traffic Type	CBR with 20 packet/sec
Packet Size	64 bytes

To evaluate the performance of the investigated QoS mechanisms, we used the following metrics:

- **Latency:** the average time in 'ms' it takes to transmit a packet from the source to the destination.
- **Jitter:** it describes how much the packets vary in latency and is determined by calculating the standard deviation of the latency.
- **Loss rate:** the loss rate determines the amount of sent packets in relation to the amount of packets that have not been received successfully at the destination.

B. Ad Hoc Routing Protocol Comparison

We have run some initial simulations to see which ad hoc routing protocol has the best performance and can be used as the starting point of our work. We investigated four different protocols, namely Ad Hoc On-Demand Distance-Vector (AODV), Dynamic Source Routing (DSR), Destination-Sequenced Distance Vector (DSDV) and Optimized Link State Routing(OLSR) [3].

With regard to latency and jitter, AODV showed the best average performance while all the other protocols showed much higher latency and jitter in the range of some hundred milliseconds. DSR's performance was by far the worst in these comparisons. Concerning loss rate, the reactive protocols (AODV and DSR) showed the best performance with less than 10 % packet loss while the proactive protocols (DSDV and OLSR) produced significantly higher losses.

Based on our initial simulation results, we selected AODV as the routing protocol of choice for our work because AODV showed the lowest latency, jitter and packet loss rate.

C. Effects of the QoS Extensions

In the following, we show and discuss our simulation results investigating the effects of the QoS extensions. Fig. 4, 5, and 6 show the impact of one QoS mechanism at a time.

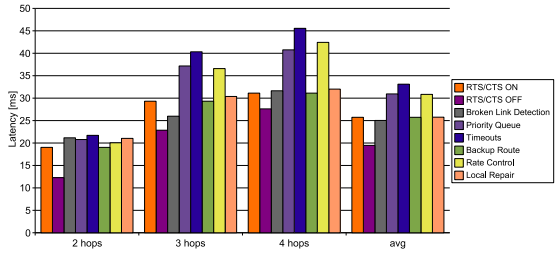

Fig. 4. Latency wrt Hop Count, and Average Latency of Game Traffic

From these figures we can see, that more or less all the QoS mechanisms, besides using RTS/CTS and local repair, improved somehow the experienced performance of the traffic, but it is hard to say clear statements about the level of the improvements. However, it is more interesting to see the combined impact of these QoS mechanisms. Thus, in the rest of the simulations we gradually extended the basic AODV protocol with the different QoS mechanisms in the order of the expected significance of the improvement caused by them and in the following figures (Fig. 7, 8, 9, 10, 11) every

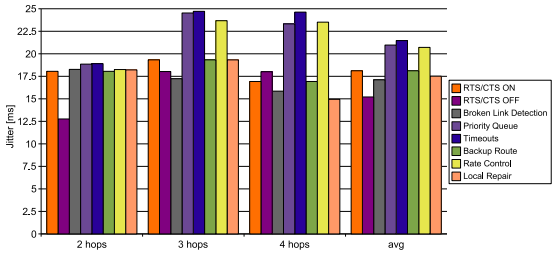

Fig. 5. Jitter wrt Hop Count, and Average Jitter of Game Traffic

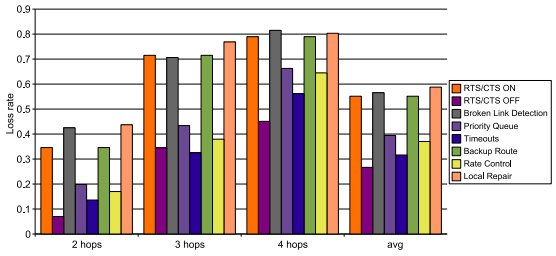

Fig. 6. Loss Rate wrt Hop Count, and Average Loss Rate of Game Traffic

bar represents the result applying its assigned plus all the preceding QoS extensions. For the evaluation we dropped all game packets with a delay of more than 100 ms because we assumed that they would not be of importance for the game anymore. With this strategy, we could reduce the average game traffic latency from approx. 300 ms to below 30 ms and the average jitter from above 50 ms to below 20 ms. Needless to say that as we dropped late packets on purpose we simply traded low latency for high packet loss. Fig. 7 and 8 show the latency and jitter experienced by the game traffic in case of different hop-number connections. As we can see from Fig. 9, we were able to reduce the game traffic loss rate significantly by introducing our QoS extensions. The average delay, jitter and packet loss rate for the background traffic are shown in Fig. 10 and 11.

Note that using local repair and RTS/CTS mechanisms degrade the overall performance in case of real-time traffic, thus we discuss them first and do not use them in the rest of the simulations. In the following, we will discuss the effects of every QoS extension in detail.

1) Local Repair: Using the local repair mechanism degrades the performance of the packet loss rate as well as latency and jitter. In particular, the average loss rate of high priority connections is increased by 5 %. First, this is due to the fact that the local repair process takes too much time for delivering real-time packets. Second, repaired routes tend to be longer than the original route and in this case the node sends an additional route error RERR message to the source which will itself restart the route discovery process again. Therefore, we do not recommend to use the built-in local repair mechanism of AODV in case of real-time traffic.

2) RTS/CTS Mechanism: The RTS/CTS mechanism degrades the performance of the routing protocol as well. The latency and jitter, and even the loss rate of the game traffic are significantly higher if RTS/CTS is enabled. The reason behind this is that multiplayer game data packets are very small in general and the RTS/CTS mechanism induces a huge overhead in this case decreasing the performance in all metrics significantly. As a result, RTS/CTS should not be activated by default and instead a threshold should be maintained by the system that depends on the number of neighbors and the size of the data packets that have to be transmitted. A neighbor list can be discovered easily by collecting MAC layer beacons from adjacent nodes.

3) Broken Link Detection: Using the Link Layer Feedback (LLF) mechanism of the MAC layer, broken links can be detected very quickly. The average latency can be reduced below 20 ms and jitter below 15 ms if LLF is activated. In addition, the packet loss rate can be reduced further by some percent as depicted in Fig. 9. Moreover, the amount of routing traffic is much smaller if LLF is used since AODV does not require HELLO messages for broken link detection any more.

4) Priority Queuing: To evaluate the effect of priority queuing, we handled the real-time game traffic as high priority traffic and the background traffic as low priority traffic. High priority packets that are not delivered within 100 ms (one-way) are regarded as lost and consequently the average latency and jitter of high priority packets are reduced significantly in all simulated scenarios, in our case to 20 ms and 14 ms. As a side effect, the packet loss rate increases as more packets are outdated. Regardless of the applied QoS mechanisms the loss rate highly depends on the number of hops. The higher the hop count, the more high priority packets are discarded since they arrive too late at the destination. However, when employing priority queuing, the loss rate of high priority packets can be substantially reduced as illustrated in Fig. 9. On the other hand, latency and jitter of low priority traffic is increased as Fig. 10 shows.

5) Timeouts: When adding timeouts to priority queuing, the loss rate of high priority traffic is slightly reduced further. Because of the small timeout value of the high priority queue, high priority packets which are late are now dropped earlier thus alleviating congestion. Connections with up to two or three hops have a loss rate of 2 % or 8 %, respectively. The average loss rate, however is still over 10 %. In addition, jitter and latency of low priority traffic can be also reduced.

6) Backup Route: The backup route mechanism increases the loss rate of high priority traffic for 2-hop connections slightly to 3 %. On the other hand, the loss rate of 3-hop connections is now reduced to 5 %. Using the backup route mechanism low priority traffic yields the lowest latency in all scenarios as depicted in Fig. 10.

7) Real-Time Neighbor Aware Rate Control: Applying our rate control mechanism the loss rate of high priority traffic has been reduced to 2 % and 4.5 % for 2-hop and 3-hop connections, respectively. Thus, 2- and 3-hop connections can be employed for real-time multiplayer games. Connections with 4 hops still suffer from a very high loss rate about 18 % and therefore cannot be used for real-time applications. Thus, the average loss rate of high priority traffic is just slightly

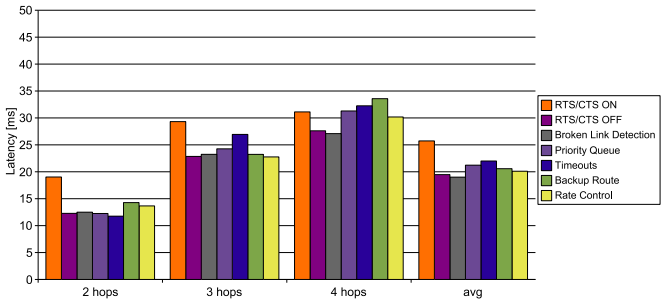

Fig. 7. Latency wrt Hop Count, and Average Latency of Game Traffic - Combined Impact

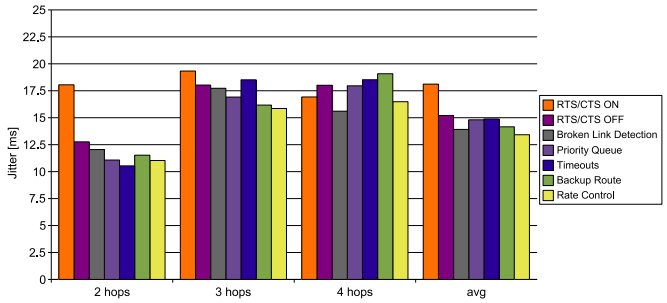

Fig. 8. Jitter wrt Hop Count, and Average Jitter of Game Traffic - Combined Impact

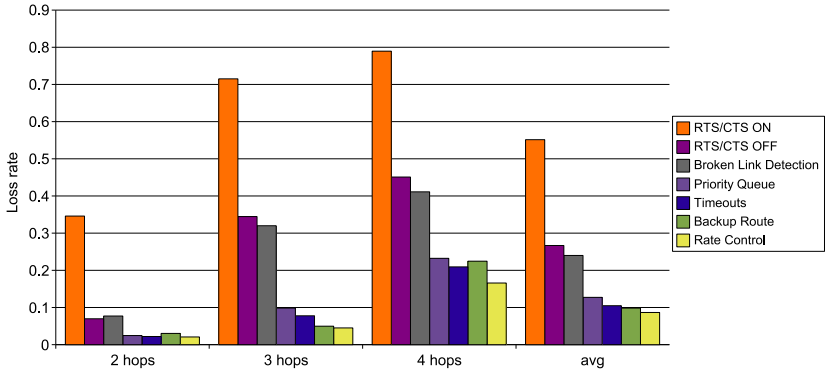

Fig. 9. Loss Rate wrt Hop Count, and Average Loss Rate of Game Traffic - Combined Impact

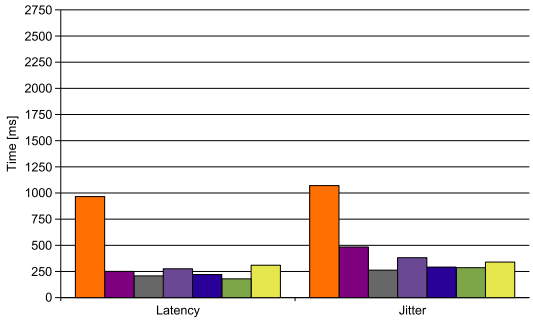

Fig. 10. Latency and Jitter of Background Traffic - Combined Impact

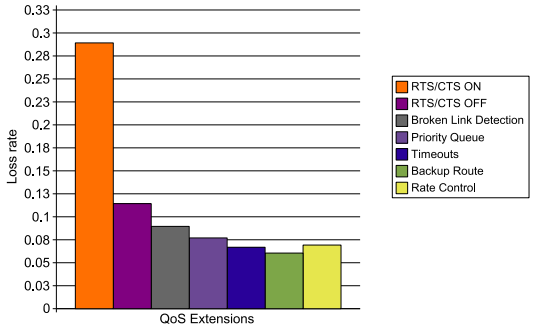

Fig. 11. Loss Rate of Background Traffic - Combined Impact

below 10 %. However, latency and jitter of low priority traffic are slightly increased since they remained in the same range in case of high priority traffic.

D. Summary of the Overall Impact of the QoS Extensions

We can see that by applying the discussed QoS extensions, except for AODV's local repair and the RTS/CTS mechanism, the packet loss rate for connections up to three hops can be reduced below 5 % since the end-to-end communication delay and delay jitter remain in the range of 25 – 30 ms and 15 – 20 ms, respectively. Moreover, the impact on the background traffic is also tolerable, i.e., the background traffic is not punished drastically in favor of the real-time traffic. Thus, other, non real-time applications can still use the network at the same time.

VI. Related Work

Many QoS architectures have been proposed for MANETs. Proposals like INSIGNIA [13], FQMM [14], or CEDAR [15] use a reservation-oriented approach and keep per-flow state information at the mobile nodes. Other approaches like SWAN [16] use a stateless feedback-based mechanism to achieve soft real-time services. All of these QoS architectures work at the network layer or above, so they make little use of information which is available at the lower layers. They also provide a self-contained solution to the QoS problem in MANETs. In this paper we take a look at individual QoS mechanisms to improve latency and jitter for real-time applications. While some of these mechanisms like priority queuing, rate control or backup route are also present in some of the QoS architectures mentioned above, mechanisms like broken link detection or

backup route belong to the routing part of the network layer and signal strength monitoring or RTS/CTS are features of the lower layers. We see this paper disjunctive to existing QoS architectures because most of the features presented here can also be applied to them.

One can find most of the QoS mechanisms mentioned in this paper also in related works.

The RTS/CTS mechanism is a common solution to the hidden/exposed terminal problem found in wireless networks and is part of the IEEE 802.11b standard [2]. As the overhead from RTS/CTS for small data packets is quite large, the standard also mentions a packet size threshold above which this mechanism should be used. Instead of just using the packet size, we proposed to also include the priority and the number of neighbors in the decision whether to enable RTS/CTS or not.

Broken link detection is part of the AODV routing protocol as defined in [17]. It also suggests that any suitable link layer notification, such as those provided by IEEE 802.11, can be used to determine network connectivity. Broken link detection with link layer feedback as well as signal strength monitoring is also used in the AODV implementation of Uppsala University [10] and is used unaltered in this paper.

In general, when looking at related work, most papers that discuss the performance of MANETs only deal with the analysis and the optimization of throughput and packet loss rather than taking latency and jitter into account. While on the other hand, papers on multiplayer games usually focus on the Internet as transport medium. In this paper, we apply existing and new QoS extensions to MANETs and compare their performance with the requirements of multiplayer games.

VII. Conclusions and Future Work

In this paper, we analyzed and evaluated common QoS extensions and proposed new approaches to mobile ad hoc networks. Following a cross-layer design, we classified these extensions into the three categories like routing protocol enhancements, traffic management mechanisms and MAC layer support mechanisms. In the simulations, we used the AODV routing protocol as the starting point of our work because AODV had shown the best initial performance compared to some other ad hoc routing protocols such as DSR, DSDV and OLSR. We have shown that, applying QoS extensions, the packet loss rate for connections up to three hops can be reduced below 5 % while the end-to-end communication delay and delay jitter remain in the range of 20 - 25 ms and 15 - 20 ms, respectively. This allows to meet the demands even of real-time multiplayer games. Moreover, we have shown how some common mechanisms like local repair of AODV and the RTS/CTS mechanism degrade the performance and should not be used in case of real-time traffic. However, as the network is getting bigger and the connections longer (consisting of more hops) giving QoS guarantees is extremely difficult without modifying the actual contention based medium access mechanism of IEEE 802.11 MAC layer.

As future work, we plan to investigate the discussed QoS extensions in a real test-bed environment. Moreover, we intend to design new mechanisms which can give protection against selfish nodes and which can provide quick channel access to nodes handling real-time traffic.

Acknowledgment

This work has been partly supported by the European Union under the E-Next NoE FP6-506869 project.

References

[1] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool, "The Effects of Loss and Latency on User Performance in Unreal Tournament 2003," in *Proceedings of the 3rd Workshop on Network and System Support for Games*, Aug. 2004, pp. 144–151.

[2] I.-S. S. Boards, "Part11-Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," 12 June 2003, http://standards.ieee.org/getieee802/802.11.html.

[3] C. E. Perkins, Ed., *Ad Hoc Networking*, 2001.

[4] Information Sciences Institute ISI, "The Network Simulator NS-2," February 2005, http://www.isi.edu/nsnam/ns.

[5] Johannes Faerber, "Network Game Traffic Modelling," in *Proceedings of the 1st Workshop on Network and System Support for Games*, Apr. 2002, pp. 53–57.

[6] N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu, "The Effect of Latency on User Performance in Warcraft III," in *Proceedings of the 2nd Workshop on Network and System Support for Games*, May 2003.

[7] G. Armitage and L. Stewart, "Limitations of using real-world, public servers to estimate jitter tolerance of first person shooter games," in *Proc. ACM SIGCHI ACE2004*, 2004, pp. 257–262.

[8] M. Dick, O. Wellnitz, and L. Wolf, "Analysis of Factors Affecting Players' Performance and Perception in Multiplayer Games," in *Proceedings of the 4th Workshop on Network and System Support for Games*, Hawthorne, USA, Oct. 2005.

[9] H. Lundgren, E. Nordström, and C. Tschudin, "Coping with communication gray zones in 802.11b based ad hoc networks," in *Proceedings of the 5th ACM international workshop on Wireless mobile multimedia (WOWMOM'02)*. New York, NY, USA: ACM Press, September 2002, pp. 49–55.

[10] Erik Nordström, "AODV-UU implementation," http://core.it.uu.se/AdHoc/AodvUUImpl.

[11] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, October 1998, pp. 85–97.

[12] T. S. Rappaport, *Wireless Communications, Principles and Practice*. PRENTICE HALL INTERNATIONAL, 1996, ISBN: 0-13-042232-0.

[13] Xiaowei Zhang and Seoung-Bum Lee and Gahn-Seop Ahn and Andrew T. Campbell, "INSIGNIA: An IP-based Quality of Service Framework for Mobile Ad Hoc Networks," in *Journal of Parallel and Distributed Computing, Special Issue on Wireless and Mobile Computing and Communications*, vol. 60. Academic Press, Inc., Apr. 2000, pp. 374–406.

[14] Xiao Hannan and Chua Kee Chaing and Seah Khoon Guan Winston, *Quality of service models for ad hoc wireless networks*. CRC Press, Inc., Dec. 2002.

[15] Prasun Sinha and Raghupathy Sivakumar and Vaduvur Bharghavan, "CEDAR: a core-extraction distributed ad hoc routing algorithm," in *Proc. IEEE INFOCOM 1999*, Mar. 1999, pp. 202–209.

[16] Gahng-Seop Ahn and Andrew T. Campbell and Andras Veres and Li-Hsiang Sun, "SWAN: Service Differentiation in Stateless Wireless Ad Hoc Networks," in *Proc. IEEE INFOCOM 2002*, June 2002, pp. 457–466.

[17] C. E. Perkins, E. M. Belding-Royer, and S. R. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," Nokia Research Center, University of California, University of Cincinnati, RFC 3561, July 2003.