A Congestion-aware Medium Access Control Protocol for Multi-rate Ad-hoc Networks

Timo Zauner[1] Luke Haslett[2,4] Wen Hu[2] Sanjay Jha[2] Cormac J. Sreenan[3]

1 Albert-Ludwigs-University of Freiburg, zauner@informatik.uni-freiburg.de

2 The University of NSW, {lah,wenh,sjha}@cse.unsw.edu.au

3 University College Cork, cjs@cs.ucc.ie

4 National ICT Australia

Abstract

This paper investigates the problem of how to improve TCP performance in multi-rate Ad-hoc networks with congested links. To improve network performance, different rate adaptation algorithms, such as Automatic Rate Fallback (ARF) and Receiver-Based AutoRate (RBAR), have been proposed to adapt the data rate according to the current channel quality. Opportunistic Auto Rate (OAR) protocol is an optimisation for any existing rate adaptation algorithm which leads to a significant performance gain by providing temporal fairness. We analyze the reasons for the high performance gain obtained using OAR, and show that the OAR protocol does not work well for TCP communications in Ad-hoc networks where nodes use different data rates to communicate with each other (heterogeneous). Based on these analysis, we propose a Congestion Reactive Opportunistic Auto Rate (CROAR) protocol, which is a new rate adaption enhancement tailored to improve TCP performance in heterogeneous multi-hop Ad-hoc networks. Extensive simulations show that CROAR, compared to OAR and RBAR, produces significant throughput and end-to-end transmission latency improvements while only marginally relaxing temporal fairness.

1. Introduction

In this paper, we investigate the problem of how to improve TCP performance, e.g., network throughput and end-to-end transmission latency, in multi-rate Ad-hoc networks with congested nodes.

The original IEEE 802.11 standard supports two data rates of 1 Mb/s and 2 Mb/s at the physical layer. Later amendments (IEEE 802.11a/b/g) support multiple higher data rates of up to 54 Mb/s depending on the channel qualities. The Auto Rate Fallback (ARF) Protocol was the first published [11] rate adaption algorithm which adjusts the data rates automatically based on channel qualities. To estimate the channel quality, ARF attempts to transmit at increasingly higher data rates after successive acknowledgement reception (and vice versa for lost acknowledgements). Receiver Based Auto Rate (RBAR) protocol [8] estimates the channel quality by measuring the signal-to-noise ratio of each received packet allowing for attempted transmission at the perceived highest possible data rate.

The Opportunistic Auto Rate (OAR) protocol allows for multiple back-to-back data packet transmission in proportion to the data rate in a manner which is temporally fair, high quality channels can transmit more packets than low quality channels yet they access the channel for the same period of time [14]. OAR is an enhancement which can be applied to any auto rate protocol such as ARF and RBAR, and demonstrates considerable throughput improvements.

However, we show through analysis and simulations that the OAR protocol has performance limitations in Ad-hoc wireless networks where nodes use different data rates between one another; in particular, when there exist congested links. We believe that congested links exist in most of the reasonably large Ad-hoc networks. Therefore, the problem of improving TCP performance of multi-rate Ad-hoc network with congested links needs to be considered. Recently, IEEE 802.11s task group has also identified congestion control as one of the to-be-solved Medium Access Control (MAC) problems in wireless mesh networks [1].

We found that nodes within data flows, that have higher quality channels to receive than to transmit, will become congested because of packets arriving faster than their abilities to forward them. OAR further deteriorates this situation, thus downgrading the performance of TCP, by allowing faster channels transfer more back-to-back packets. This observation motivates our work on multi-hop Ad-hoc networks with congested nodes. We propose a *Congestion Reactive Opportunistic Auto Rate (CROAR)* protocol which

1-4244-0419-3/06/$20.00 ©2006 IEEE

overcomes the limitations of OAR by allowing the transfer of more back-to-back packets on congested channels. Extensive simulations show that CROAR, compared to OAR and RBAR, produces significant throughput and latency improvements while only marginally relaxing temporal fairness.

The rest of this paper is organized as follows. Section 2 discusses related work in this area. In Section 3, we outline the the design of the Congestion Reactive Opportunistic Auto Rate (CROAR) Protocol, and explain both its congestion reactive and symmetric mechanisms in details. We evaluate CROAR with extensive simulations in Section 4. Section 5 gives conclusions and describes areas for future work.

2. Related Work

The IEEE 802.11, also known as Wi-Fi (Wireless Fidelity), is a set of standards for Wireless Local Area Networks (WLAN). The main scope of the IEEE 802.11 standards are to develop a MAC and Physical layer (PHY) specification for wireless connectivity for fixed, portable and moving stations within a local area [9]. It describes the functions required by a wireless device to operate within a wireless network. The original standard was published in 1997 and supported data rates of 1 and 2 Mb/s. Since then several amendments were proposed, such as 802.11a/g which supports maximum data rates of up to 54 Mb/s and 802.11b of up to 11 Mb/s. However, the IEEE 802.11 standard does not propose a rate adaptation algorithm to take advantage of these higher data rates. It is the responsibility of those who implement IEEE 802.11 device drivers to design and implement a rate adaptation algorithm which adjusts the data rate according to the channel quality. As shown in different publications [7, 8, 11, 12, 4, 14], the use of rate adaptation at the MAC layer can significantly improve the performance of wireless networks. Several rate adaptation algorithms have been proposed such as *Auto rate fallback* (ARF), *Receiver-Based Auto Rate* (RBAR).

ARF [11] was the first published commercial rate adaption algorithm. The original proposal [11] was designed for the original IEEE 802.11 standard and provided rate adaption for data rates of 1 Mb/s and 2 Mb/s only, it can easily be extended to other amendments however. The data rate is determined by gathering statistics on the success and loss of acknowledgements at each sender, increasing when the sender receives acknowledgements for a certain number of consecutive packets and vice versa for losses.

RBAR [8] uses the channel quality information available at the receiver (signal-to-noise ratio) in order to select a data rate. This approach takes into account that the channel quality might not be symmetric. Therefore, only the receiver can estimate the channel quality accurately. After estimating the channel quality and selecting the best data rate according to a lookup table, the receiver must communicate the data rate back to the sender. The sender can then transmit the following packets using the new data rate.

2.1. The OAR Protocol

The Opportunistic Auto Rate (OAR) protocol [14] is an enhancement for any existing IEEE 802.11 rate adaptation algorithm. It works by exploiting wireless channels whenever the channel quality is good, allowing higher sending rates to be used. As OAR is an optimisation for rate adaptation algorithms, it firstly needs a rate adaptation algorithm on the MAC layer which selects a sending rate according to the current channel quality, e.g., ARF, RBAR, etc. Secondly, OAR needs a mechanism to hold the channel for an extended period of time whenever it detects a high quality channel in order to exploit it. The exploitation of high quality channels can be accomplished by providing temporal fairness rather than packet fairness which is provided by existing IEEE 802.11 wireless networks [2]. Packet fairness implies that every node can transmit one single packet whenever it gains access to the channel. Temporal fairness implies that every node is granted the same medium access time to transmit packets. The number of back-to-back packets a node is able to transmit at a time can be computed as follows:

$$number\ of\ packets = \left\lfloor \frac{transmission\ rate}{base\ rate} \right\rfloor \qquad (1)$$

Therefore, in IEEE 802.11b wireless networks with a base rate of 2 Mb/s, nodes with a transmission rate of 11 Mb/s are able to send 5 packets; nodes with a transmission rate of 5.5 Mb/s are able to send 3 packets; and nodes with a transmission rate of 2 Mb/s (base rate) are able to send one packet whenever they gain access to the channel.

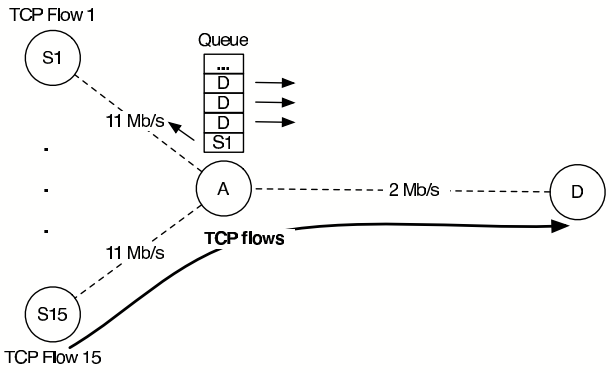

Figure 1. Simulation scenario

OAR Performance Limitations: An overview of the multi-hop OAR simulation scenario from Sadeghi et al [14]

can be found in Figure 1. The distances between nodes S_1 through S_{15} and node *A* are sufficiently small such that a data rate of 11 Mb/s is available. Nodes *A* and *D* are sufficiently far apart so that the data rate of 2 Mb/s is available. There is one TCP flow from each node S_1 through S_{15} to node *D*. In order to study the TCP throughput as a function of the queue size at node A, the interface queue size at nodes S_1 through S_{15} and node *D* is set to 50 packets and the queue size at node *A* is variable. Note that node *A* represents a congested node as the bandwidth of all incoming links is 11 Mb/s whereas the bandwidth of the outgoing link to node *D* is only 2 Mb/s. OAR compounds this problem further due to its temporal fairness property, each of nodes S_1 through S_{15} will transmit 5 packets back-to-back to node *A* causing its queue to fill more rapidly.

The OAR simulations from Sadeghi et al contain an implementation error which overshadows this limitation. Nodes have one interface queue, between the link layer and MAC layer. When a node communicates with one other node, all packets in the interface queue are addressed to the same destination. In our example from Figure 1 however, node *A* communicates with nodes S_1 through S_{15} and node *D*, meaning the interface queue can contain packets to different destinations.

Using Figure 1, a snapshot of the interface queue at node *A* shows the next packet in the queue is addressed to node S_1 (TCP acknowledgement). As the link between node *A* and S_1 allows a data rate of 11 Mb/s, OAR is able to send 5 back-to-back packets. However, the next packet in the queue is addressed to node *D*. This is where the original OAR implementation contains an error. Rather than sending one packet only to node S_1, it sends out the following 4 packets to *D* back-to-back when normally it would be limited to a single packet at 2 Mb/s. It does not include a check for whether the following packets in the interface queue are addressed to different nodes. Further, no separated RTS/CTS handshake is performed for these following packets destined for node *D* causing a further violation. This error can be solved by each node recording from which node it received its last CTS message and for all following back-to-back packets to check whether the destinations of these packets are equal to the sender of the last CTS.

In general, the temporal fairness provided by OAR is not beneficial for heterogeneous multi-rate multi-hop scenarios. Heterogeneous means that the links in a multi-hop path supply different data rates. Nodes that have higher quality channels to receive than to transmit can become congested because of packets arriving faster than their abilities to forward them. Heterogeneous multi-hop networks could contain many of these nodes. These nodes have an increased risk of congestion if using OAR, as it allows faster channels to transfer more back-to-back packets than slower channels. This observation motivates our work on multi-hop Ad-hoc networks with congested nodes.

3. CROAR Protocol

As shown in the previous section, the OAR protocol does not work well for heterogeneous multi-hop wireless scenarios. Here we present the Congestion Reactive Opportunistic Auto Rate (CROAR) MAC protocol. Similar to OAR, it is an enhancement that can be applied to any MAC auto rate protocol (e.g., ARF, RBAR, etc.) with the same basic temporal fairness properties as OAR. Unlike OAR, it will allow a node suffering from congestion to use more temporal blocks to send additional packets back-to-back, allowing it to reduce its interface queue length. Using additional temporal blocks create asymmetric problems within flows, hence CROAR also contains a mechanism to promote symmetry within flows.

3.1. Congestion Reactive Mechanism

The purpose of the Congestion Reactive Mechanism of the CROAR protocol is to provide temporal fairness in scenarios with low traffic load and relax temporal fairness in scenarios with high traffic load to reduce the probability of packet loss in the network. In highly traffic-loaded scenarios the queues of congested nodes will increase to capacity and overflow eventually. Therefore, in order to minimize queue overflows, as the queue occupancy at a node increases, more packets are allowed to send back-to-back once the node gains access to the channel.

The OAR protocol provides temporal fairness by allowing each node to access the channel for the same amount of time, which we refer to as a temporal block. CROAR adapts the number of temporal blocks a node is granted to access the channel by monitoring a nodes congestion. As a given nodes queue occupancy increases and crosses specified thresholds, it is allowed to use additional temporal blocks to send more packets back-to-back. Table 1 shows how many packets can be transmitted for each combination of temporal block and IEEE 802.11b data rate. OAR represents a single temporal blocks allowing transmissions of 1, 3 and 5 packets for data rates of 2, 5.5 and 11 Mb/s respectfully. In CROAR, if a given congested node is allowed 3 temporal blocks to reduce its level of queue occupancy, it will be able to transmit up to 3, 9 and 15 packets back-to-back for data rates of 2, 5.5 and 11 Mb/s respectively.

We next define an algorithm to determine how many temporal blocks to assign a node depending on its current queue occupancy level compared to a series of defined thresholds. Let τ_i denote the thresholds with $\tau_0 < \ldots < \tau_N$; $\tau_N = 100\%$. Let $Q_{ifq}(t)$ denote the queue occupancy at time t. The number of temporal blocks a node is granted

Temporal Blocks	2 Mb/s	5.5 Mb/s	11 Mb/s
1	1 packet	3 packets	5 packets
2	2 packets	6 packets	10 packets
3	3 packets	9 packets	15 packets
...	...	...	...

Table 1. Number of back-to-back packets a node is able to send for different data rates

to access the channel can be computed using a linear function $f_{lin}(t)$ or an exponential funtion $f_{exp}(t)$:

$$\begin{array}{lll} f_{lin}(t) = i+1 & Q_{ifq}(t) \leq \tau_i & \text{i} = 0, \ldots, \text{N} \\ f_{exp}(t) = 2^i & Q_{ifq}(t) \leq \tau_i & \text{i} = 0, \ldots, \text{N} \end{array} \quad (2)$$

Table 2 shows the results of applying the above algorithm. For congestion thresholds of 25%, 50%, 75% and 100%, a linear configuration will be allowed to use 1, 2, 3 and 4, and an exponential configuration 1, 2, 4 and 8 temporal blocks to transmit packets. Exponential configuration should be preferred in highly loaded scenarios as it enforces a more aggressive queue occupancy reduction than the linear configuration.

Queue Occupancy	$f_{lin}(t)$	$f_{exp}(t)$
$Q_{ifq}(t) \leq 25\%$	1	1
$Q_{ifq}(t) \leq 50\%$	2	2
$Q_{ifq}(t) \leq 75\%$	3	4
$Q_{ifq}(t) \leq 100\%$	4	8

Table 2. Number of temporal blocks a node is granted using the CROAR protocol

3.2. Symmetric Mechanism

TCP provides reliability by requiring an acknowledgement after data packet transmission. Slow delivery of these acknowledgements can have significant impact on TCP performance. The TCP sliding window mechanism allows a TCP sender to send a specified number of packets in the sliding window, after which the node remains idle waiting for acknowledgements to allow it to advance the window and transmit further packets. Additionally, if a TCP acknowledgement is not received within a certain period of time, the TCP retransmission timer (RTO) times out causing a retransmission of its associated data packet. For these reasons it is clear that TCP acknowledgements should be forwarded as fast as possible in order to achieve the highest TCP performance and avoid asymmetry.

A problem with the proposed CROAR protocol when used with TCP is that it provides performance improvement in only one direction of the TCP flow, i.e., from the TCP sender to the TCP receiver. If a node becomes congested and is allowed more temporal blocks to send additional data packets back-to-back, its next hop neighbour also needs to be able to return an equal number of acknowledgements; otherwise, an imbalance and asymmetry will result (see Section 4 and Figure 3). In order to provide symmetric improvements in both directions of the TCP flow, each node must be able to send the same number of back-to-back packets to a neighbour as it previously received from that neighbour.

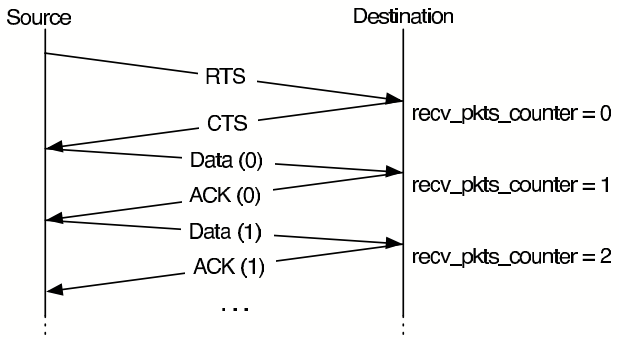

Figure 2. Counting number of back-to-back packets received

This idea can be implemented using a lookup table. In order to populate and maintain the table, each node must count the number of back-to-back packets most recently received from each of its neighbours and store the value in the lookup table. Therefore a counter `recv_pkts_counter` is introduced which is set to zero whenever a node sends a CTS. Whenever a node sends a MAC ACK, the `recv_pkts_counter` is increased by one (Figure 2). Then, before transmitting a packet to a neighbour, the node must first check how many packets it previously received from that neighbour in the lookup table. If the value is greater than the number of back-to-back packets determined by the CROAR protocol, the node uses this value to determine the number of back-to-back packets it is allowed to send.

4. Evaluation

In this section, we evaluate CROAR by state-of-the-art discrete event network simulator *ns-2* [5].

4.1 Simulation Environment

The support of TCP, the IEEE 802.11 MAC layer and multiple data rates on the physical layer are important for our simulations. By default *ns-2* does not provide rate adaptation algorithms on the MAC layer. In order to investigate rate adaptation algorithms in IEEE 802.11b networks, we started our work using the 2.1b7 code base. In addition, we installed a rate adaptation extension available from the Rice Networks Group. The extension contains implementations of the RBAR and OAR rate adaptation algorithms and is available from the CMU Monarch Project website[1]. The available data rates for RBAR and OAR are set to 2 Mb/s, 5.5 Mb/s and 11 Mb/s according to the IEEE 802.11b standard. In order to provide temporal fairness, OAR is configured to send 1 packet at a data rate of 2 Mb/s, 3 back-to-back packets at 5.5 Mb/s and 5 back-to-back packets at 11 Mb/s. Two ray ground propagation model is used to simulate the wireless channel.

On the data link and physical layer, the IEEE 802.11 standard is used. The default parameters used for both the IEEE 802.11 MAC and PHY layer are set according to the IEEE 802.11 standard. The transmission range for a data rate of 2 Mb/s is set to 250 meters, for a data rate of 5.5 Mb/s to 200 meters and for a data rate of 11 Mb/s to 100 meters. On the link layer, each node has a drop-tail queue which holds 50 packets unless specified otherwise. Packet transmission is scheduled according to the First-In, First-Out (FIFO) principle. On the network layer, Destination-Sequenced Distance Vector (DSDV) protocol [13] is used as the routing protocol for Ad-hoc wireless networks. TCP Reno is chosen on the transport layer as it is the de-facto standard used in most TCP implementations. The maximum size of the TCP congestion windows is set to 20 packets and the TCP packet size is set to 1,000 bytes. On application layer, File Transfer Protocol (FTP) is used to generate traffic over a TCP connection. The start time of the simulations is set to 200 seconds in order to allow the routing algorithm DSDV sufficient time to build all routing tables. All simulations were run 20 times for 50 seconds, and the presented results were computed as the average of these multiple simulations.

4.2. Throughput

Figure 3 shows the throughput of the original OAR implementation for the scenario described in Figure 1. The results obtained are similar to the results presented in the original OAR publication [14]. However, the OAR results obtained, after resolving the implementation error described in Section 2.1, differ significantly. Rather than increasing the throughput by approximately 30% compared to RBAR, the

[1] http://www-ece.rice.edu/networks/ext/oar/

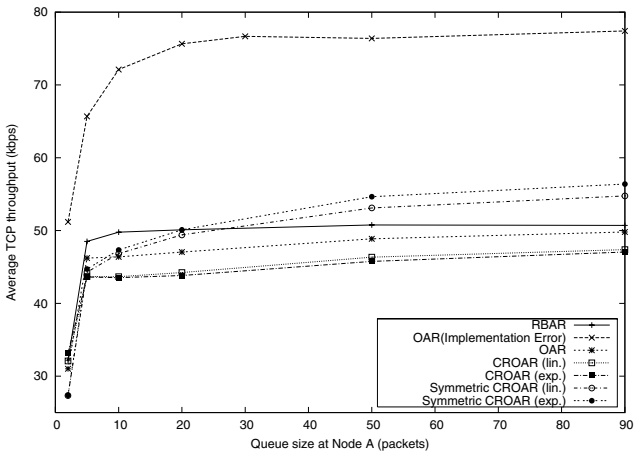

Figure 3. Throughput of original OAR protocol and CROAR mechanisms

throughput obtained using OAR is just under the throughput obtained with RBAR. Figure 3 also shows the resulting throughput performance with each of the new schemes. The configuration of the CROAR protocol we used for our simulations can be found in Table 2. We divided the queue into four equal parts using thresholds of $\tau_0 = 25\%$, $\tau_1 = 50\%$, $\tau_2 = 75\%$ and $\tau_3 = 100\%$.

It can be observed that both asymmetric CROAR configurations result in a throughput decrease compared to RBAR. Although this result is counterintuitive, it highlights the need to provide an additional mechanism to preserve symmetry in TCP communication when dynamically altering the number of back-to-back packets transmitted, justifying the development of symmetric CROAR. Figure 3 also shows that the symmetric CROAR protocol significantly improves the throughput compared with asymmetric CROAR. With an increasing queue occupancy at node A (greater than 20 packets) the symmetric CROAR using the exponential configuration results in a performance gain of up to 12% compared to RBAR and of up to 13% compared to the original OAR protocol.

4.3. Latency

Figure 4 shows the end-to-end transmission latency for TCP data packets of each individual TCP flow. Both asymmetric CROAR schemes perform worse than RBAR and OAR due to its introduced asymmetry between nodes *A* and *D*. As the queue occupancy at node *A* increases and crosses the specified thresholds in table 2, it consumes more temporal blocks for node *A*'s transmissions, decreasing node *D*'s ability to access the medium to transmit TCP acknowledgements. However, both symmetric CROAR schemes demonstrate latency reductions of 20%-25% when compared to

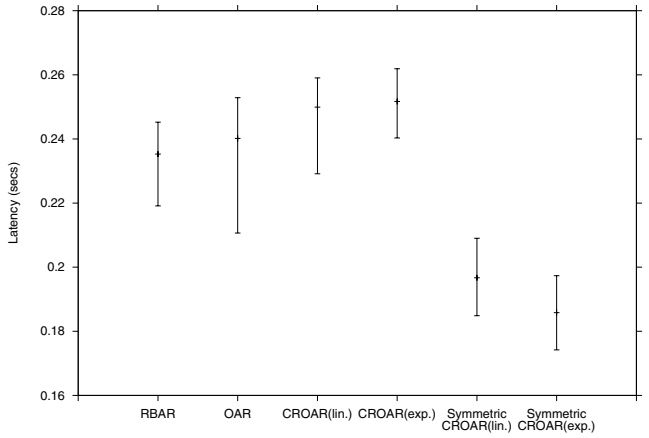

Figure 4. Latency of individual TCP flows

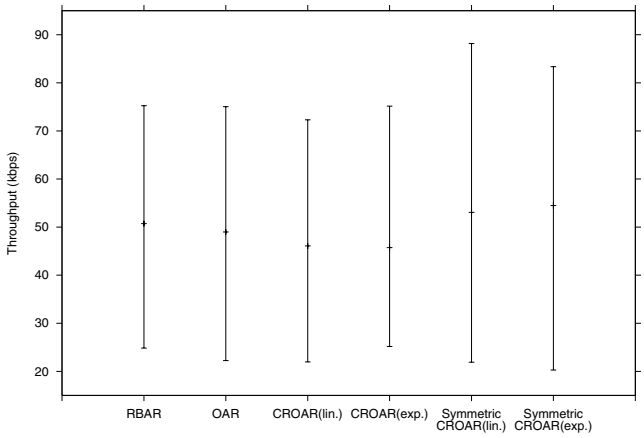

Figure 5. Fairness of individual TCP flows

RBAR. The reason for this is node *A* will be forced to spend more time receiving back-to-back TCP acknowledgements from node *D*, reducing the number of RTS/CTS handshakes required. Additionally, as these TCP acknowledgements will be back-to-back, TCP data packets will not be able to arrive from the source nodes at node *A* in between, as they can in the other schemes, reducing the amount of time these TCP data packets spend in the network.

4.4. Fairness of TCP flows

OAR ensures temporal fairness, i.e., all nodes are allowed to access the channel for the same amount of time. The temporal fairness scheme results in different throughput values for channels with different data rates. The goal of this experiment is to investigate the throughput fairness of the rate adaptation protocols for single TCP flows from nodes S_1 through to S_{15} to node *D* in the scenario pictured in Figure 1.

Figure 5 shows the minimum, maximum and mean throughput achieved for all single TCP flows. It can be observed that the throughput achieved by the symmetric CROAR varies slightly more than for the other schemes. However, as it also provides a throughput gain of approximately 10% to 15% compared to RBAR, OAR and the asymmetric CROAR, the fairness results are still within an acceptable range.

4.5. Multiple TCP flows and throughput

In the simulation scenario above, each TCP sender/receiver pair communicates using one TCP flow only. For a more realistic approach, it is important to also research the performance of multiple TCP flows per TCP sender/receiver pair. Therefore, we investigated the performance obtained for 50 TCP flows per TCP sender/receiver pair as follows.

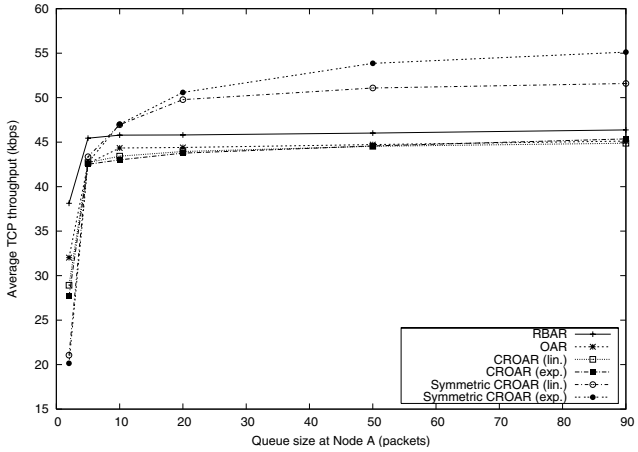

Figure 6. TCP throughput for 50 TCP flows

Figure 6 shows the performance of RBAR, OAR and the proposed CROAR improvements for multiple TCP flows. It can be observed that for 50 TCP flows, the symmetric CROAR protocol results in a throughput gain of 18% compared to RBAR and 22% compared to OAR. The throughput gain with a growing number of TCP flows is due to the fact that the symmetric CROAR protocol is the only protocol which not only increases the number of back-to-back packets for TCP data, but also the number of back-to-back packets for TCP ACKs. These results imply that the symmetric CROAR is particularly well suited for highly loaded scenarios with multiple TCP flows per TCP sender/receiver pair.

4.6. TCP window size and throughput

The relationship between the maximum TCP congestion window size *MaxWin* and the throughput in multi-hop wireless networks have been previously researched [6, 10]. As a result, it was shown, that the default *MaxWin* is too high for wireless multi-hop scenarios. It was also shown that the *MaxWin* value resulting in a maximum TCP throughput can be determined by a function of the number of hops traversed by the TCP flow. In order to confirm these results and to research the impacts of *MaxWin* on CROAR, we simulated the scenario shown in Figure 1 for different *MaxWin* values.

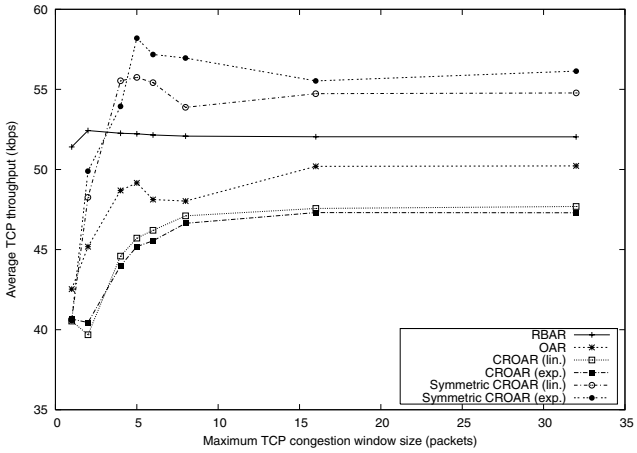

Figure 7. Impact of $MaxWin$ on the TCP throughput

Figure 7 shows the simulation results for a variable setting of $MaxWin$. It shows that the performance results do not differ significantly for the different *MaxWin* settings after *MaxWin* is larger than a small number of packets (e.g., 5). We decided to use the default value of *MaxWin* (20 packets) for the remaining simulation experiments.

4.7. Two-hop chain topology

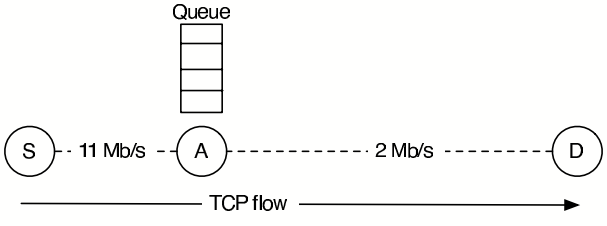

Figure 8. Two-hop chain topology for multi-hop TCP experiments

In this scenario the TCP throughput will be researched for a two-hop chain topology as shown in Figure 8. Nodes *S* and *A* are sufficiently close together such that the data rate is 11 Mb/s and nodes *A* and *D* are sufficiently far apart such that the data rate is 2 Mb/s. There is one TCP flow from node *S* to node *D*.

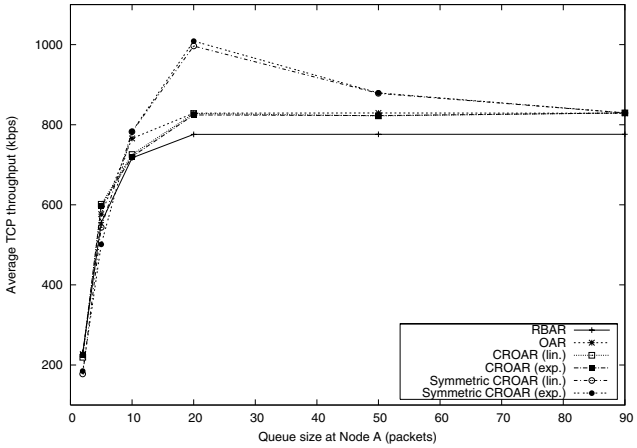

Figure 9. Results for two-hop chain topology with one TCP flow

Figure 9 shows the average throughput as a function of the queue size at node *A* for one TCP flow from node *S* to node *D*. Firstly, it can be observed that in this scenario, OAR results in a throughput gain compared to RBAR. For the asymmetric CROAR and symmetric CROAR, it can be observed that the linear and the exponential configuration result in nearly identical results. The most interesting result of this scenario is the fact that there is a maximum in the throughput of the symmetric CROAR for a queue size of approximately 20 packets. The reason for this behaviour can be found in the low amount of traffic in this scenario. For bigger queue sizes, the queue will not fill up and as a consequence CROAR will not trigger the use of multiple back-to-back packets for the bottleneck link.

Figure 10 shows the results for the same two-hop chain topology with 50 flows between node *S* and node *D*. It can be observed that as a consequence of the traffic increase, both configurations of the symmetric CROAR result in a throughput gain for all queue sizes. The exponential configuration of the symmetric CROAR results in a throughput gain of approximately 23% compared to the original OAR protocol and of 31% compared to RBAR.

5. Conclusion and Future work

In this paper, we found that the performance of the OAR protocol has deficiencies for multi-rate multi-hop Ad-hoc

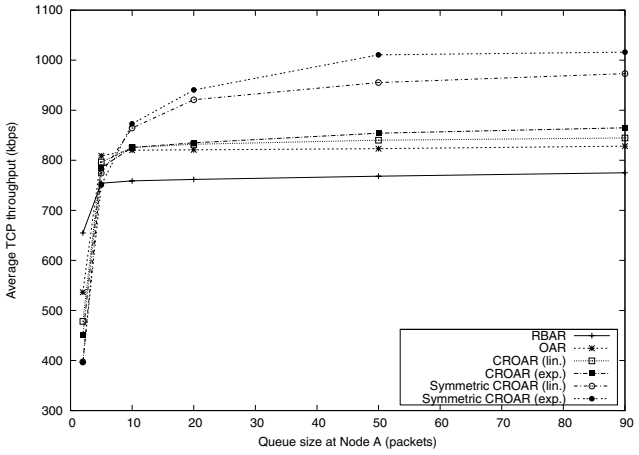

Figure 10. Results for two-hop chain topology with 50 TCP flows per TCP sender/receiver pair

wireless networks, specifically in heterogeneous networks with varying transmission data rates (Ad-hoc network with congested nodes). To improve performance in these networks, we proposed the CROAR protocol. We showed by using extensive simulations that CROAR offers significant throughput and latency gains when compared with RBAR and OAR, while only slighty reducing fairness.

For scenarios with low traffic load we observed that symmetric CROAR only increases the throughput for small queue sizes. As a consequence, the idea of using the queue occupation as an indication of a congesting node does not work in scenarios with large queue sizes and a low load of traffic. An idea for a better approach to detect congesting nodes is to keep track of the relation between incoming and outgoing packets. If there are more incoming than outgoing packets, a bottleneck is detected and the number of outgoing packets should be increased in order to increase the overall performance.

In the example in which we discovered the OAR implementation error we observed that consecutive packets in the interface queue can be addressed to different nodes. This fact prevents OAR from sending the maximum number of back-to-back packets. So another idea to improve the performance of the OAR protocol is to ensure that OAR is always able to send the maximum number of back-to-back packets for a given data rate. One way to achieve this is to search the interface queue for packets addressed to the same destination. Alternatively, instead of using one interface queue at each node, multiple queues could be used, e.g., one queue for each destination.

References

[1] Joint SEE-Mesh/Wi-Mesh Proposal to 802.11 tgs, Feb. 2006.

[2] B. Awerbuch, D. Holmer, and H. Rubens. Effects of Multi-rate in Ad Hoc Wireless Networks. Technical report, Johns Hopkins University, 2003.

[3] S. Bansal, R. Shorey, and A. Kherani. Performance of TCP and UDP Protocols in Multi-Hop Multi-Rate Wireless Networks. In *IEEE Wireless Communications and Networking Conference (WCNC'04)*, 2004.

[4] J. del Prado Pavon and S. Choi. Link Adaptation Strategy for IEEE 802.11 WLAN via Received Signal Strength Measurement. In *IEEE International Conference on Communications (ICC '03)*, volume 2, pages 1108–1113, Anchorage, Alaska, USA, May 2003.

[5] K. Fall and K. Varadhan. The ns manual. *The VINT Project*, 2000.

[6] Z. Fu, H. Luo, P. Zerfos, S. Lu, L. Zhang, and M. Gerla. The Impact of Multihop Wireless Channel on TCP Performance. In *IEEE Transactions on Mobile Computing*, volume 4, pages 209 – 221, March-April 2005.

[7] I. Haratcherev, K. Langendoen, R. Lagendijk, and H. Sips. Hybrid Rate Control for IEEE 802.11. In *MobiWac '04: Proceedings of the second international workshop on Mobility management & wireless access protocols*, pages 10–18, New York, NY, USA, 2004. ACM Press.

[8] G. Holland, N. Vaidya, and P. Bahl. A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 236–251, New York, NY, USA, 2001. ACM Press.

[9] IEEE Computer Society. IEEE Std 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, June 1997.

[10] R. Jiang, V. Gupta, and C. V. Ravishankar. Interactions between TCP and the IEEE 802.11 MAC protocol. In *DARPA Information Survivability Conference and Exposition*, volume 1, pages 273 – 282, 2003.

[11] A. Kamerman and L. Monteban. WaveLAN II: A High-Performance Wireless LAN for the Unlicensed Band. *Bell Labs Technical Journal*, pages 118–133, 1997.

[12] M. Lacage, M. H. Manshaei, and T. Turletti. IEEE 802.11 Rate Adaptation: A Practical Approach. In *MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 126–134, New York, NY, USA, 2004. ACM Press.

[13] C. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[14] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. Opportunistic Media Access for Multirate Ad Hoc Networks. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 24–35, New York, NY, USA, 2002. ACM Press.