Alleviating MAC Layer Self-Contention in Ad-hoc Networks

Zhenqiang Ye[†], Dan Berger[†], Prasun Sinha[‡], Srikanth Krishnamurthy[†], Michalis Faloutsos[†], Satish K. Tripathi[†]

[†] Dept. of Computer Science and Engineering
UC Riverside
{zye, dberger, krish, michalis, tripathi}@cs.ucr.edu

[‡] Dept. of Computer and Information Science
Ohio State University
prasun@cis.ohio-state.edu

I. Introduction

The distributed coordination function (DCF) mode of IEEE 802.11 has become the *defacto* standard media access control mechanism for wireless ad-hoc network research. By design the IEEE 802.11 MAC protocol is unaware of the transport layer connection a packet belongs to. As a result packets belonging to the same connection contend for local spectra during transmission at neighboring nodes. This phenomenon, termed **self-contention**, is one of the key reasons for significantly lower goodput over multi-hop connections in wireless ad-hoc networks. In this article we propose two MAC layer mechanisms to alleviate self-contention and, consequently, contention in general.

We loosely define a *stream* as the sequence of packets from a specific source node to a specific destination node. So, in a single TCP connection, the TCP-DATA packets constitute one stream and the TCP-ACK packets constitute a second stream. We distinguish between two types of self-contention, intra-stream and inter-stream. Intra-stream self-contention is caused by packets of the same stream competing for the shared medium. Contention caused by TCP-DATA packets on other TCP-DATA packets is an example of intra-stream contention. In transport protocols such as TCP, RTCP, or SCTP, a reverse stream typically carries acknowledgments or feedback information for controlling the forward stream. The contention caused by the packets of the reverse stream on the packets of the forward stream, or vice versa, is called inter-stream contention. For example, the contention caused by TCP-ACK packets competing with TCP-DATA packets is inter-stream contention.

We find that self-contention should be resolved at the MAC layer for the following three reasons. First, self-contention arises due to distributed access of the shared media which is the role of the MAC layer. Second, widely deployed upper layer protocols, such as UDP and TCP, are left unchanged. Third, the MAC protocol commonly used for multi-hop networks, namely IEEE 802.11, is an evolving standard, and is therefore more amenable to enhancements than transport layer protocols.

Previous work addresses this problem only partially or at layers other than the MAC. To the best of our knowledge, we are the first to address inter-stream self-contention in ad-hoc networks at the MAC layer. Inter-stream contention has been studied in the context of wireless LANs [4], where an approach similar to quick-exchange is proposed. Another solution at the routing layer was proposed to reduce the interaction of opposing streams [2]. For reducing intra-stream contention, [3] proposes a link layer mechanism and [5] proposes a transport layer solution. Our proposed changes reside only at the MAC layer - they neither require nor preclude solutions at other layers.

To reduce the effect of self-contention, we propose two MAC layer mechanisms; *quick-exchange* and *fast-forward*. The *quick-exchange* mechanism targets inter-stream self-contention by attempting to subsume the contention caused by the reverse stream of the transport connection on the forward stream. The *fast-forward* mechanism targets intra-stream self-contention by attempting to withhold the transmission of a packet at the sender until the previous packet of the same stream has reached beyond the interference range of the sender.

By reducing self-contention, our mechanisms increase the performance of the network. We observe a significant improvement in the throughput for both TCP and UDP. We trace the cause of this improvement to reduced MAC layer control overhead, reduced number of false-link-failures, and decreased time spent in back-off state between packet transmissions.

II. Our MAC Enhancements

A. Quick-Exchange

Quick-Exchange[1] provides an efficient mechanism for exchanging two data packets between adjacent nodes in the same dialogue (RTS-CTS exchange) as shown in Figure 1[2]. The standard RTS-CTS-DATA1-ACK1 dialogue is extended by an additional data packet transmission (DATA2) from the RTS-receiver. The DATA2 packet contains the piggybacked acknowledgment (ACK1) for DATA1. If DATA2 is received correctly, the RTS-sender sends a corresponding acknowledgment (ACK2).

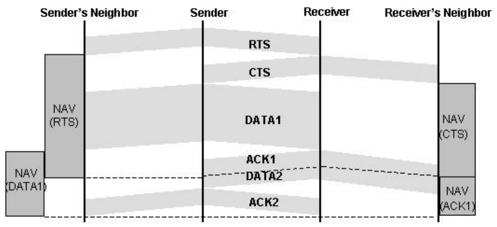


Fig. 1. Quick-Exchange

Quick-Exchange obviates RTS and CTS transmissions for DATA2. The intuition behind Quick-Exchange is to make use of the fact that two nodes have agreed to communicate at a given time. It also eliminates the idle time due to the back-off prior to the transmission of DATA2 if the IEEE 802.11 MAC is used. Transmission of DATA2 is free of channel contention and this reduces the incidence of false link failures while improving end-to-end goodput.

Our implementation of Quick-Exchange addresses several subtle issues. We present the modification that ensure that all nodes in the neighborhood are aware of the extended length of the

[1] A detailed study of quick-exchange is available in [1].

[2] Details of inter-frame spacings, such as SIFS and DIFS, are omitted from the figures and the discussion for clarity of presentation.

communication. Consider the Sender and Receiver as the communicating pair and their corresponding neighbors in Figure 1. The Sender sends an RTS as usual. The Receiver replies with a CTS containing an additional field that indicates the extra duration needed for successful transmission of DATA2. Increasing the communication duration advertised in the CTS is not desirable, as it leads to wasted capacity in case the transmission of CTS or DATA1 fails. So the receiver's neighbors become aware of the extended communication on reception of ACK1 and not the CTS. Thus, Sender's neighbors are warned of the extended communication

Note that the DATA1 and DATA2 packets need not be from the same transport connection. In addition, we advocate its use for packet exchanges where at least one of the two packets is a small packet (like a TCP-ACK) to avoid excessively long capture of the channel.

B. Fast-Forward

Fast-forward attempts to forward a packet as soon as it is received. This prevents the sender from inserting packets into the network before the previous packet of the same stream has traveled beyond its interference range.

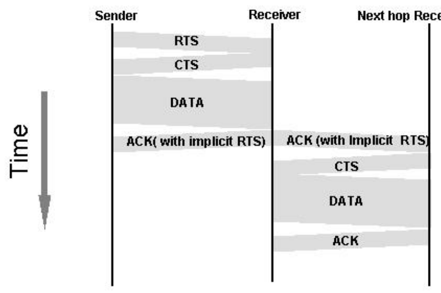


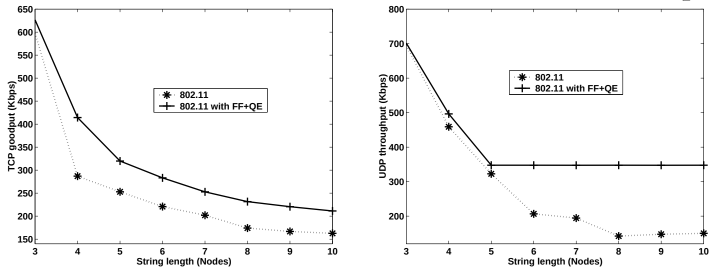
Fig. 2. Fast-Forward

After a packet is received at a node, the receiver determines the next-hop for the packet and uses the MAC layer ACK as an implicit RTS for the next hop. The sequence of packet transmissions during fast-forwarding is shown in Figure 2. This requires addition of "RTS destination" and "source address" fields in the ACK packet. The former is used to identify the next hop node, and the latter is needed for the next hop node to respond with a CTS. Fast-forwarding saves an RTS packet for the forwarded transmission. In addition, the forwarded transmission is not preceded by any backoff time, and this increases the channel utilization. The reduced contention due to fast-forwarding reduces the number of false link failures and improves end-to-end goodput.

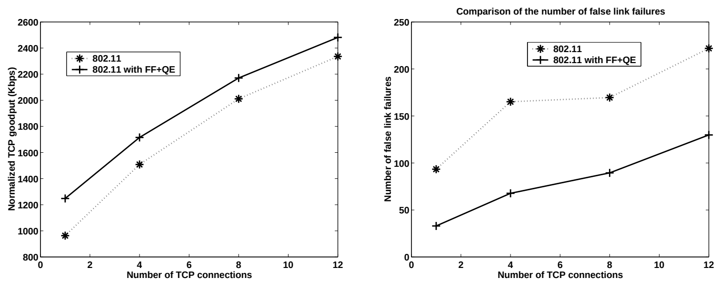
III. Performance Evaluation

We study the performance of the combination of the two approaches using *ns2* and show some preliminary results in Figure 3. Most papers use aggregate TCP goodput as a performance metric; note however that this metric can be easily increased by favoring smaller flows over longer flows, as the latter requires more spectrum. We use the normalized TCP goodput, which is obtained by a weighted sum of the goodputs of individual TCP flows where the hop lengths are used as the weights.

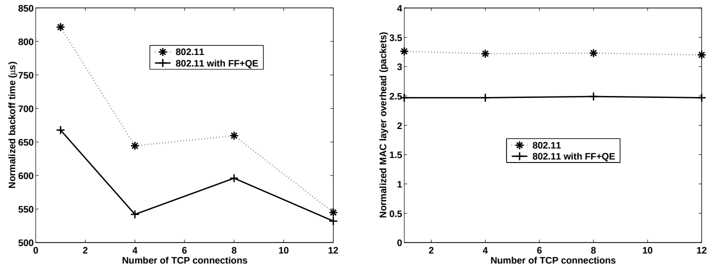
Our MAC enhancements can increase the TCP goodput by as much as 45% (4 node string topology) and UDP goodput by



(a) TCP traffic, string topology (b) UDP traffic, string topology



(c) TCP traffic, random topology (d) Number of false link failures (random topology)



(e) Average backoff time per data packet (random topology) (f) Average MAC layer control overhead per data packet (random topology)

Fig. 3. 802.11 vs. 802.11 with our enhancements

250% (strings with more than 7 nodes) in string topologies. In random topologies, the goodput for TCP flows improves by up to 30%. There are three key reasons for the goodput increase. First, our enhancements reduce the number of false link failures by as much as 66%. Second, the average backoff time per data packet is reduced by up to 19%. Third, the MAC layer control packet overhead is reduced from 3.2 to 2.47 control packets per data packet.

IV. Conclusions

We propose two MAC layer enhancements to address self-contention in ad-hoc networks. The quick-exchange enhancement reduces inter-flow self-contention and the fast-forward enhancement reduces intra-flow self-contention. Preliminary studies on the *ns2* simulator show significant improvement in goodput over the original IEEE 802.11 MAC scheme.

References

[1] D. Berger, Z. Ye, P. Sinha, S. Krishnamurthy, M. Faloutsos, and S. K. Tripathi. Alleviating MAC Layer Self-Contention in Multi-hop Wireless Networks. Technical Report, Dept of CS, UC Riverside, 2003.
[2] C. Cordeiro, S. R. Das, and D. P. Agrawal. COPAS: Dynamic Contention-Balancing to Enhance the Performance of TCP over Multi-hop Wireless Networks. In *Proc. 10th Intl. Conf. on Computer Comm. and Networks (IC3N)*, pages 382–387, October 2002.
[3] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla. The Impact of Multihop Wireless Channel on TCP Throughput and Loss. In *Proc. of IEEE INFOCOM*, 2003.
[4] C. Parsa and J. J. Garcia-Luna-Aceves. Improving TCP Performance over Wireless Networks at the Link Layer. *ACM Mobile Networks and Applications Journal*, 5(1):57–71, 2000.
[5] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar. ATP: A Reliable Transport Protocol for Ad-hoc Networks. In *Proc. ACM MOBIHOC*, June 2003.