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 and, 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, TCP-DATA packets constitute one stream and TCP-ACK packets constitute a second. 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. Transport protocols such as TCP, RTCP, and SCTP, use a reverse stream to carry 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 inter-stream contention. An example is the contention caused by TCP-ACK packets competing with TCP-DATA packets.

We find that self-contention should be resolved at the MAC layer for three reasons. First, self-contention is caused by distributed access to the shared media - precisely the role of the MAC layer. Second, a MAC layer solution leaves widely deployed upper layer protocols, such as UDP and TCP, unchanged. Third, the MAC protocol commonly used for multi-hop networks, namely IEEE 802.11, is an evolving standard, and 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. A routing layer solution was proposed in [2] to reduce the interaction of opposing streams. 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 modifications 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 stream has traveled beyond the interference range of the sender.

By reducing self-contention, our mechanisms improve network performance. We observe a significant improvement in goodput 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 a piggybacked acknowledgment (ACK1) for DATA1. If DATA2 is received correctly, the RTS-sender sends a corresponding acknowledgment (ACK2).

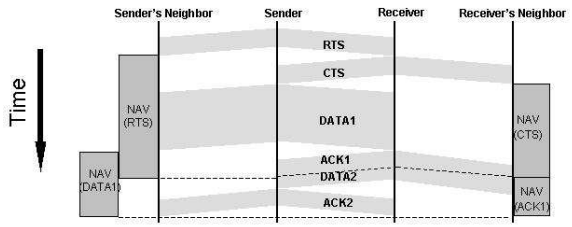


Fig. 1. Quick-Exchange

The intuition behind Quick-Exchange is to leverage the fact that two nodes have agreed to communicate at a given time and hold a reservation on the channel. Quick-Exchange obviates RTS and CTS transmissions for DATA2 and eliminates the back-off time required by IEEE 802.11 prior to the transmission of DATA2. Transmission of DATA2 is free of channel contention which reduces the incidence of false link failures while improving end-to-end goodput.

Our implementation of Quick-Exchange addresses several subtle issues. We detail here the mechanism which ensures that all neighboring nodes become aware of the extended communication duration. Consider the sender, receiver and their corresponding neighbors, as in Figure 1. The sender sends an RTS indicating 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.

duration required for the transmission of DATA1[3]. The receiver replies with a CTS containing an additional field indicating the extra duration needed for transmission of DATA2. Note that increasing the duration advertised in the CTS is not desirable, as it leads to wasted channel capacity if either the CTS or DATA1 are not successfully transmitted. The sender's neighbors are notified of the extended channel reservation by the increased duration advertised in DATA1, and the receiver's neighbors are notified on receipt of ACK1/DATA2.

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 (such as a TCP-ACK) to avoid excessive channel capture.

B. Fast-Forward

Fast-forward attempts to forward a packet immediately upon receipt. This prevents the sender from inserting packets into the network before the previous packet of the stream has traveled beyond its interference range.

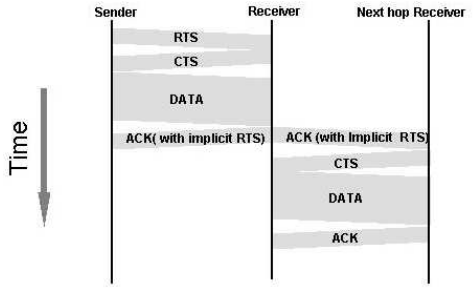


Fig. 2. Fast-Forward

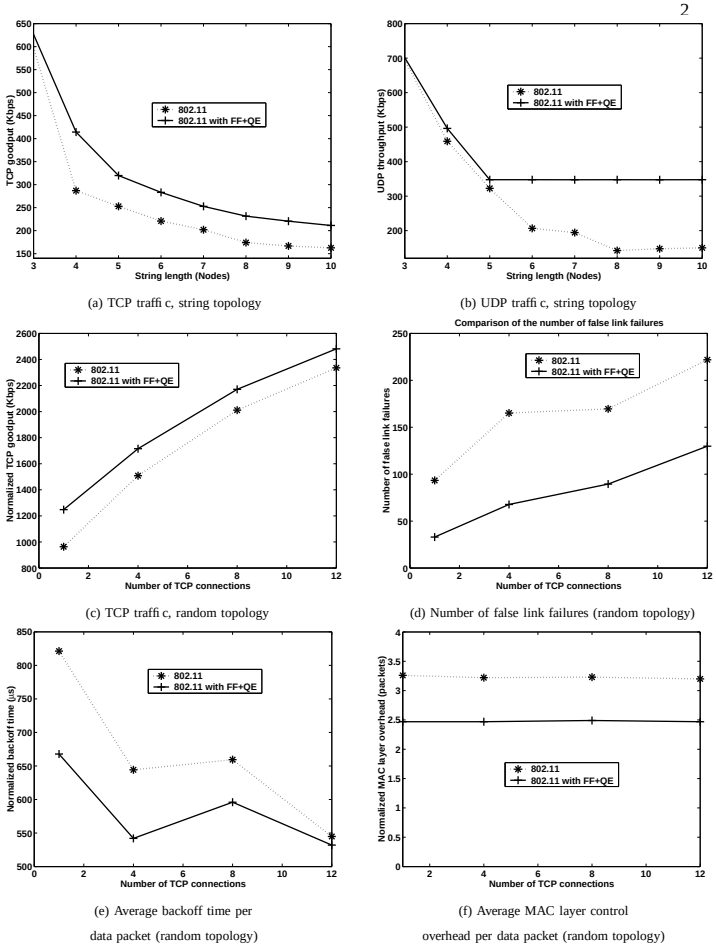
When a packet is received at a node, the receiver determines the next-hop for the packet and uses the MAC layer ACK as an implicit RTS for the next hop. The sequence of packet transmissions during fast-forwarding is shown in Figure 2. This requires addition of "RTS destination" and "source address" fields 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 avoids the 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; however we note that this metric can be easily skewed by favoring smaller flows over longer flows, as the latter require more spectrum. We instead use normalized TCP goodput, a weighted sum of the goodputs of individual TCP flows using the connection length (in hop count) as the weights.

Our MAC enhancements can increase the TCP goodput in string topologies by as much as 45% (4 node string) and UDP goodput by 250% ($>= 7$ nodes). In static random topologies, the goodput of 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.

[3]and associated control packets



(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

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.