Improving the Performance of Multi-hop Wireless Networks using Frame Aggregation and Broadcast for TCP ACKs

Wonsoo Kim Hyrum K. Wright Scott M. Nettles

Wireless Networking and Communications Group (WNCG)
Department of Electrical and Computer Engineering
The University of Texas at Austin
1 University Station C0803, Austin, TX 78712-0240
Email: {wkim, hwright, nettles}@ece.utexas.edu

ABSTRACT

As data rates supported by the physical layer increase, PHY and especially MAC overheads increasingly dominate the throughput achievable by wireless networks. A promising approach for reducing these overheads is to aggregate a number of frames together into a single transmission. The 802.11n standard uses such an approach for unicast frames. We present the design of a system that can aggregate both unicast and broadcast frames. Further, the system can classify TCP ACK segments so that they can be aggregated with TCP data flowing in the opposite direction. A novel aspect of our work is that we implement and validate our design not through simulation, but rather using our wireless node prototype, Hydra, which supports a high performance PHY based on 802.11n. Our validation shows significant improvements in throughput for each kind of aggregation we support.

Categories and Subject Descriptors

C.2.2 [**Computer Communication Networks**]: Network Protocols; C.1.2 [**Network Architecture and Design**]: Wireless Communication

General Terms

Design, Protocols

This material is based in part upon work supported by the National Science Foundation under grants EIA-0322957, CNS-0435307, and CNS-0626797, the Air Force Research Lab under grant numbers FA8750-06-1-0091, and FA8750-05-1-0246, the Office of Naval Research under grant number N00014-05-1-0169, and the DARPA IT-MANET program, Grant W911NF-07-1-0028.

ACM CoNEXT 2008, December 10-12, 2008, Madrid, SPAIN

Keywords

transmission control protocol (TCP), frame aggregation, 802.11n, wireless multi-hop networks

1. INTRODUCTION

As the demand for high data rates increases, wireless networking systems are deploying broadband communication high-throughput technologies such as orthogonal frequency division multiplexing (OFDM) and multi-input multi-output (MIMO). These technologies allow the data portions of frames to be transmitted at high data rates, which decreases the time spent transmitting data, but does not generally decrease the time spent on a variety of overheads. These include the time spent waiting to gain access to the transmission floor, exchanging control frames required by the MAC protocol, and physical layer (PHY) headers. The result is that these overheads begin to dominate performance even when the PHY is capable of high data rates. In general, this problem becomes more severe as rate increases because the time to transmit the data decreases, but the transmit time for most of the overheads does not. Similarly, the effect of these overheads is more dominant for short frames, such as those typically used for control, than for longer ones.

One approach to reducing these overheads and thus achieving the potential performance gains offered by modern PHYs is to group (or *aggregate*) several frames together into one transmission. This has two benefits: one, it reduces the total number of transmissions, resulting in less time waiting for the floor and transmitting control frames, and two, it reduces header overhead by allowing several frames to share headers. As an example of this approach, the IEEE 802.11n standard includes several frame aggregation schemes to support high data rates as part of its high throughput MAC design [8].

Most frame aggregation schemes require that frames that are aggregated all be destined to the same receiver. This approach neglects the fact that transmissions are broadcast and a single transmission will potentially be

received by many receivers. A simple way of taking advantage of this is to aggregate broadcast frames along with a group of unicast frames all destined for one receiver. Because broadcast frames do not require acknowledgement, this can be done while still having a single ACK for the unicast frames.

We expect TCP traffic to be important for wireless networks and it can also benefit from frame aggregation. The key observation is that TCP ACKs are small packets that flow in the opposite direction from the (typically) larger TCP data packets. Since TCP ACKs are cumulative and thus carry redundant information, they have lower reliability requirements than the data packets. We take advantage of this by treating TCP ACKs as if they were broadcast frames and do not require link-level acknowledgements. This allows the ACKs to be aggregated with TCP data traveling in the opposite direction, significantly reducing the cost of the TCP ACKs. By allowing the MAC to investigate TCP headers, our design breaks layering abstractions and thus is a cross-layer algorithm.

We present a design, implementation, and performance evaluation of three aggregation techniques: unicast aggregation, broadcast aggregation, and TCP ACK aggregation. Although our design is a general modification of 802.11, our implementation is specifically for our Hydra wireless node prototype [6], which supports an 802.11n-based PHY, with OFDM and MIMO. Thus our performance evaluation is based on a real operational wireless network, rather than simulation.

Our paper is organized as follows. Section 2 outlines existing aggregation schemes. Section 3 details the design of our protocol, while Section 4 describes its implementation on the Hydra prototype. Sections 5 and 6, presents our experimental setup and performance evaluation of our approach. We discuss our conclusions and future enhancements in Section 7.

2. RELATED WORK

Despite the wide variety of physical layer approaches to increasing wireless network throughput, MAC overhead ultimately limits maximum achievable throughput [21]. Further, these limits have more impact for frames with small data payloads and the impact increases as the rate used for data increases. Frame aggregation can help to address both the problem of short frames and of overhead becoming dominant at high data rates. Here we describe previous efforts to improve throughput using frame aggregation.

To improve throughput, the IEEE 802.11n [8] high throughput standard adopts two approaches to frame aggregation: the aggregated MAC service data unit (A-MSDU), and the aggregated MAC protocol data unit (A-MPDU) [1]. A-MSDU aggregates packets from the upper layer and adds a single MAC header and checksum. This scheme is effective when the MAC aggregates many small user packets such as TCP ACKs or other control oriented data. A-MPDU concatenates normal 802.11 MAC frames each having its own MAC header and checksum. Each of these subframes is separated by a MAC delimiter, which includes a length, checksum, and delimiter signature. The MAC delimiter allows a receiver to robustly separate each subframe, even in the case where some errors occur in the individual subframe. This approach has more overhead than the first, but supports a block ACK scheme. Block ACKs allow each subframe to be acknowledged separately, thus allowing retransmission of only the subframes in error. This approach will have an advantage with high error rates. Kim et al. [12] evaluate the throughput of an early variant of 802.11n frame aggregation as a function of payload size and physical data rate.

The 802.11n MAC also specifies a bi-directional data transfer method that can reduce floor acquisition overhead [1]. It is particularly useful for reducing the overhead of a bi-directional stream of TCP data and ACKs. When a node transmits a frame, instead of relinquishing the floor when the transmission completes, the node can grant the receiver permission for a reverse direction transmission destined for the original transmitter. This approach allows both TCP data and ACKs to be transmitted in turn. This saves a floor acquisition time and the time to exchange a request-to-send (RTS) and clear-to-send (CTS) if they are being used. However, this method does not reduce the MAC and PHY header overheads or the cost of link-level ACKs.

Skordoulis et al. [18] proposed a two-level frame aggregation scheme that mixes 802.11n's two aggregation methods. In the first stage, the MAC aggregates user packets from the upper layer into an A-MSDU with a MAC header and checksum. Then, a series of these A-MSDUs are concatenated into an A-MPDU with each A-MSDU separated by a MAC delimiter. This scheme increases the maximum aggregation size compared to using A-MSDUs and reduces MAC header overheads compared to using A-MPDUs. It allows the block ACK scheme to be applied to the A-MSDUs.

Kim et al. [11] proposed a multi-layer scheme that provides aggregation at both the MAC and PHY layers. The MAC aggregates multiple MAC frames into an A-MPDU, and then the PHY aggregates a series of A-MPDUs into a single physical frame. Within the physical frame, an additional physical delimiter precedes each of the A-MPDUs. The physical delimiter contains modulation and coding scheme information for each A-MPDU, and thus allows each A-MPDU to be transmitted at a different rate. Unlike the other existing approaches, this scheme also supports multi-destination aggregation because each A-MPDU can be addressed to a different destination. To facilitate this, the protocol

employs a polling scheme so that frames sent to different destinations can be acknowledged.

Sadeghi et al. [16] proposed the opportunistic auto-rate (OAR) method, which uses frame aggregation to take advantage of favorable channel conditions. When the underlaying rate adaptation algorithm shows that a frame can be sent at higher than base-rate, the MAC attempts to aggregate frames so that the time spent sending the frame at the higher rate equals the time that would be the same as the time to send a single frame at base-rate. This preserves the basic fairness capabilities of the 802.11 MAC while taking advantage of higher rates and the overhead reduction of frame aggregation.

There have been cross-layer approaches to improve TCP performance by piggybacking small TCP ACKs with link-level frames [14, 19, 20, 17]. Parsa et al. [14] proposed the transport unaware link improvement protocol (TULIP) that provides a piggyback method which transfers a TCP ACK with a link-level frame. Tourrilhes [19] proposed PiggyData of which idea is to transmit PiggyData ACK, a link-level acknowledgement with a flag indicating status of transmit queue, as a response of TCP data. Setting the flag to one means that a TCP ACK is followed after SIFS interval. Xiao [20] suggested a piggyback mechanism which allows the MAC to piggyback a link-level acknowledgement with a TCP ACK in a single frame immediately after the node receives TCP data. Our scheme differs from these approaches in that these do not reduce the MAC and PHY header overheads or the cost of link-level ACKs.

Scalia et al. [17] suggested PiggyCode which allows the MAC to combine TCP data with TCP ACK by using network coding technology. This approach can reduce the MAC and PHY header overheads for TCP ACKs, similar to our approach. However, the PiggyCode requires additional header to encode and decode packets, and it allows only packets of different types to be coded together, which restricts transmitting a single TCP data and TCP ACK at a time.

3. DESIGN

Our design incrementally extends the familiar IEEE 802.11 distributed coordination function (DCF) MAC protocol [7] to support frame aggregation. This addition requires modest changes to the PHY frame format, demonstrating one of the cross-layer aspects of our design. Extensions include support for unicast aggregation, broadcast aggregation, and link-level classification of TCP ACKs as broadcast frames.

3.1 Unicast Aggregation

The 802.11 standard includes significant overheads when transmitting a single frame, as shown in Table 3 and 4. These overheads include gaining access to the floor, PHY and MAC header overheads, and control frame

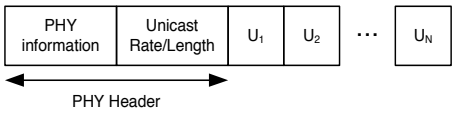

Figure 1: Unicast Aggregation Format

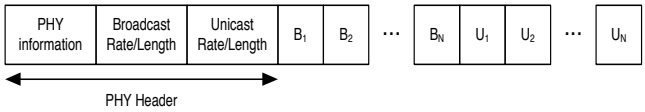

Figure 2: Broadcast Aggregation Format

overheads for RTS, CTS, and ACK frames. Unicast aggregation reduces the impact of this overhead by combining frames being transmitted to the same destination, similar to what the IEEE 802.11 standard describes. In addition to reducing header overheads, aggregating frames allows us to reduce the total number of transmissions, and thus amortizes much of the per-frame overhead over several frames.

Figure 1 is the format for supporting unicast aggregation, where U_N denotes the N-th unicast subframe. The aggregated frame consists of some PHY-oriented information, such as training sequences, the rate and length fields for the frame and then a series of unicast subframes, all bound for the same destination. Because all the unicast subframes are destined for the same node, a single ACK can be used to acknowledge all the subframes. Here, no changes are needed to the PHY.

3.2 Broadcast Aggregation

Broadcast frames are likely to be important in a multihop wireless network, especially for control protocols. For example, the dynamic source routing and ad-hoc on-demand distance vector routing protocols use broadcast frames for route discovery and maintenance [9, 15]. The broadcast nature of radio frequency transmissions means broadcast frames can not only be aggregated with each other, but also with unicast frames. This promises to significantly lower the impact of flooding-based control protocols on data transport.

Figure 2 shows how our broadcast aggregation format extends the basic unicast format, where B_N stands for the N-th broadcast subframe. Our design modifies the PHY header to add a rate and length field for the broadcast subframes. The rate information enables us to support different data rates for broadcast and unicast subframes. The length information allows our protocol to prepend a variable number of broadcast subframes to a variable number of unicast subframes, all within the same physical frame. Our design requires modifying the PHY header to include rate and length information to allow the receiving PHY to decode incoming streams. The broadcast subframes are not acknowl-

edged and thus a single link-level ACK is still sufficient. In addition, depending on queue status, the broadcast aggregation scheme allows the MAC to aggregate broadcast or unicast frames only.

3.3 Treating TCP ACKs as Broadcasts

Many of the Internet's most important applications use TCP as the transport protocol. Thus, breaking layer abstractions and optimizing for TCP becomes important. Furthermore, TCP represents a general class of protocols that support reliable transmission. TCP relies upon a bi-directional traffic flow of TCP data and TCP ACKs. Because the ACKs are small, the impact of the fixed overhead becomes even more significant, making them especially good candidates for aggregation.

TCP employs a cumulative acknowledgement mechanism. In this scheme, receipt of an ACK for packet P_i, where i is the sequence number of the packet, implies acknowledgement of all previous packets P_j, for $j \leq i$. Because of this redundancy, ACKs have less need for reliable transmission than data. Indeed, some TCP implementations intentionally drop some fraction of the ACKs to reduce protocol overhead [2]. This suggests that TCP ACKs could be transmitted without link-level acknowledgement, in the same manner as broadcast frames.

Our design breaks layer boundaries in an novel way by classifying TCP ACKs as broadcast frames and then aggregating them in the same manner as frames with broadcast addresses. This can potentially cut the number of transmissions and thus floor acquisitions needed by a TCP flow in half as well as save the significant other overheads associated with transmitting the small TCP ACK frames.

TCP ACK aggregation does not require a new frame format. Instead, TCP ACKs are categorized as link-level broadcasts and transmitted in the broadcast portion of the frame. Although the TCP ACKs are transmitted as a broadcast subframe and thus do not generate link-level ACKs, they still have unicast MAC addresses. When a node receives a TCP ACK not addressed to it, it drops the packet, rather than passing it up the stack. This behavior is significant; if the packet was passed up the stack to the IP layer, it would attempt to deliver the packet, resulting in improper duplication of the TCP ACK. Thus, instead of broadcasting TCP ACKs to the entire network, as a typical broadcast packet, the ACKs are just broadcast within the range of the nodes along the TCP stream's path, just as they would be if the ACKs were sent as unicast packets.

4. IMPLEMENTATION

We implemented this protocol using Hydra, our wireless node/network prototype. In this section, we overview Hydra and present the details of our implementation.

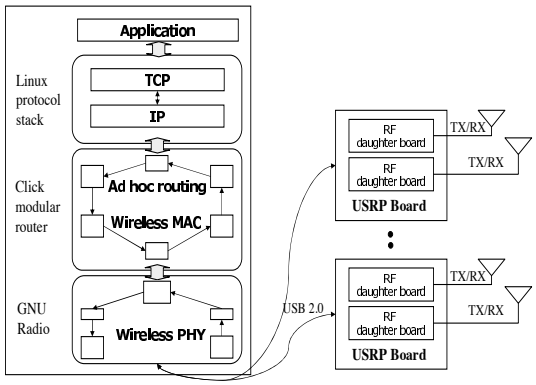

Figure 3: Block diagram of a Hydra node

4.1 Hydra Background

Hydra is a wireless network prototype being developed at the University of Texas at Austin [6]. Hydra was designed to allow both the PHY and the MAC to be flexible and easy to modify. This design allows us to experiment with cross-layer designs, such as the one presented here, on realistic hardware and real RF channels rather than just in simulations.

Figure 3 presents a block diagram of the main components of the Hydra including the RF front-end, the PHY, and the MAC. The programmable RF front-end is the universal software radio peripheral (USRP) [3], which interfaces to the general purpose host through a USB 2.0 connection. Figure 3 shows several USRPs with multiple antennas. Using multiple antennas enables us to exploit the MIMO capabilities of the protocols we implement. All other aspects of Hydra, the PHY, MAC, and higher layers, run on a general purpose computer running Linux. In addition to the MAC, we implement ad-hoc routing using Click, which interfaces with the Linux TCP stack [13]. This allows us to use standard network software for experiments.

4.1.1 PHY

The Hydra physical layer essentially follows the IEEE 802.11n standard [8] and is implemented in the GNU Radio framework [4]. This open-source software allows developers to implement signal processing blocks in C++ and then flexibly connect them together using Python as a glue language. The PHY uses BPSK, QPSK, 16-QAM and 64-QAM as its modulation schemes and $\frac{1}{2}$, $\frac{2}{3}$, $\frac{3}{4}$ and $\frac{5}{6}$ as the rates of its convolutional encoder. It supports various MIMO transmission modes including beamforming, spatial multiplexing, and cyclic delay diversity. For the experiments described here, we only use cyclic delay diversity. In addition to the standard 802.11n features, the PHY includes a link adaptation algorithm using explicit feedback. Hydra's data rates are limited due to the bandwidth of the USB and

Table 1: Hydra PHY Details

System Bandwidth	1 MHz*
Center Frequency	$2.4 - 2.5$ GHz
Maximum TX Power	10 mW
Modulation	BPSK, QPSK, 16-QAM, 64-QAM
Coding	Bit-Interleaved Binary Convolutional
SISO Data Rates	0.65*, 1.30*, 1.95*, 2.60*, 3.90*, 5.20*, 5.85*, 6.50* $Mbps$
MIMO Data Rates	$2\times$, $3\times^{+}$, and $4\times^{+}$ SISO Data Rates
Diversity Schemes	Cyclic Delay Diversity, Space-time Coding, Spatial Mapping, Beamforming

* indicates non-standard values
+ indicates capabilities in development

processing delay created by the software implementation of the PHY. Thus the prototype supports physical layer data rates 10 times less than the actual data rates defined in the IEEE 802.11n standard. Table 1 summarizes the features of the Hydra's physical layer.

4.1.2 MAC

The MAC is written in C++ using the Click modular router framework [13]. This software framework, developed at MIT, runs on a general purpose processor and was originally created for building flexible and high performance routers. Similar to GNU radio, Click allows users to build packet processing elements in C++ and connect them using its own glue language.

The Hydra MAC follows the IEEE 802.11 MAC standard for DCF with a RTS/CTS exchange. In addition, the Hydra supports an explicit feedback scheme using the RTS/CTS exchange and rate adaptation schemes including receiver based auto rate (RBAR) and auto rate fallback (ARF) [5, 10]. The present design and experiments do not use the rate adaptation schemes.

4.2 Aggregation

We enhanced the Hydra MAC and PHY to support unicast, broadcast, and TCP ACK aggregation as described in Section 3. We describe the MAC subframe format used by our aggregation schemes, the receive and transmit processes, and finally details of the classification of TCP ACKs as broadcasts.

4.2.1 Frame

A single physical frame includes a series of MAC subframes. These subframes are embedded in the aggregated frame shown in Figure 2. Figure 4 shows the format of each MAC subframe. This follows the standard 802.11 MAC format with the exception that we eliminated the address 4 field because we do not support infrastructure networking. Each subframe includes a MAC header containing general information: duration, source and destination addresses, and length. Our aggregation protocol only uses the duration field of the first unicast subframe for virtual carrier sensing. However, for the purpose of easy prototyping, all of the subframes have the duration field. Each subframe includes a 2-byte length field. Finally all of the subframes contain frame check sequence (FCS) and PAD octets. Future work includes eliminating the redundant information currently appearing in the headers.

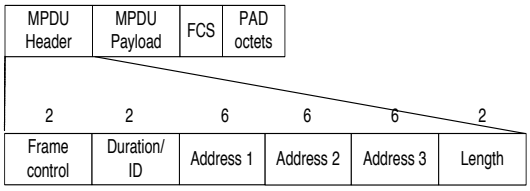

Figure 4: The MAC subframe format.

Frames transmitted in the broadcast portion of the frame can have a broadcast or unicast address but are not acknowledged. On the other hand, the subframes transmitted in the unicast portion require an ACK and thus all must be addressed to the same destination.

4.2.2 The Receive Process

When receiving a frame, the PHY uses the broadcast length and rate information to decode the broadcast subframes and then the unicast length and rate information to decode the unicast subframes. Once the PHY completes decoding all the subframes, it sends the subframes up to the MAC. When the MAC receives an aggregated frame, it first processes the broadcast subframes and then processes the unicast subframes. For the broadcast portion, as soon as each subframe passes the cyclic redundancy check (CRC), the MAC sends the subframe to the next layer. Thus the broadcast subframes do not suffer from higher loss probability though they are aggregated with unicast subframes. For the unicast subframes, the MAC checks destination address and all of the CRCs, and, if they all pass, then the MAC sends them up to the next layer and sends a link-level ACK. Otherwise, all of the unicast subframes are discarded. We could optimize by storing and applying CRCs to aggregates instead of individual subframes. However, the current scheme has only a small overhead and will allow us to extend our design to a block ACK scheme like that of 802.11n.

4.2.3 The Transmit Process

On the transmit side, the MAC must assemble the aggregated frames into the correct format. To achieve this, we have two queues: One for broadcasts and one for unicasts. The MAC first searches the broadcast queue and assembles all the broadcast frames. Then the MAC searches the unicast queue and gathers the unicast frames being transmitted to the same destination as the first frame in the unicast queue. Once completed, the MAC aggregates the broadcast subframes followed

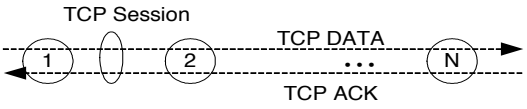

Figure 5: Linear topology with one TCP session

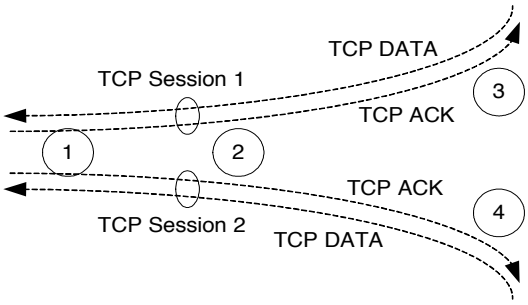

Figure 6: Star topology with two TCP sessions

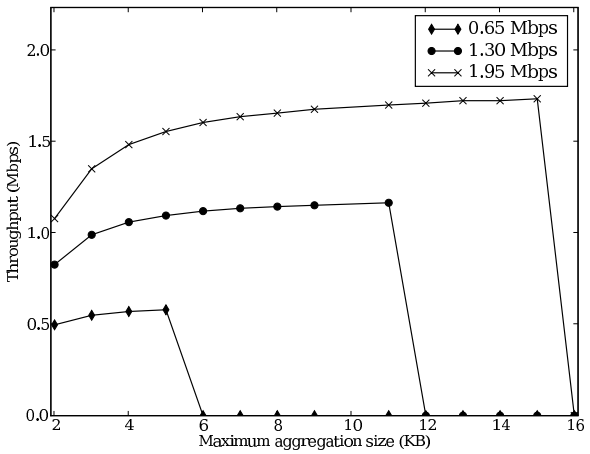

Figure 7: Throughput vs. Aggregation size

by unicast subframes up to a parameterized maximum aggregation size. Putting the broadcasts ahead of the unicasts enables the broadcasts to be less sensitive to changes in the wireless channel. This is because the channel might change during transmission and the subframes close to the PHY training sequences are less likely to be corrupted by these changes. Once the frames are assembled, the MAC hands the aggregated frame down to the PHY along with the rate and length information for the broadcast and unicast parts of the frame. The entire transmit process triggers when the DCF of the MAC acquires the floor.

4.2.4 TCP ACKs

The process above neglects TCP ACKs, which are specially handled when assigning packets to the unicast or broadcast queues. We assign "pure" TCP ACKs to the broadcast queue. We define "pure" TCP ACK segments to be those that do not contain any data and are not part of connection set-up. Click provides a packet classification mechanism, and our implementation uses these classifiers to sort pure TCP ACKs from other unicast frames and place them in the broadcast queue.

5. EXPERIMENTAL SETUP

For our experiments, we equipped our USRP frontends to operate in the 2.4 GHz frequency band and we used a transmission power of 7.7 mW which ensured a signal-to-noise ratio (SNR) level of 25 dB for the node spacing we used. This SNR did not allow reliable operation of the rates that required 64-QAM. We used the cyclic-diversity MIMO mode and thus transmitted a single data stream from our two antenna systems. We ultimately did experiments at 0.65, 1.30, 1.95, and 2.60 Mbps. Higher rates would have shown greater improvements for our techniques, but the rates used serve to validate our concept.

We used three different topologies: 2- and 3-hop linear topologies as shown in Figure 5; and a star topology as shown in Figure 6. Spacing between the nodes is roughly 2.5 meters. Although Click provides several ad-hoc routing protocols, because all of our nodes are within transmission range of the others, they would have not discovered any multi-hop routes. Instead we used static routing to force the topologies in the figures.

To create UDP traffic we used an application that simply sent UDP packets at a controllable rate. For our experiments, we sent UDP packets that resulted in 1140B MAC frames. For TCP, we used a one-way file transfer to send a 0.2 Mbyte file. The maximum segment size for TCP was set to 1357 bytes, resulting in a MAC frame of 1464 bytes. TCP ACKs had MAC frame sizes of 160 bytes.

6. EXPERIMENTAL RESULTS

We performed a series of experiments to study the impact of frame aggregation for unicast frames, broadcast frames, and TCP ACKs. To begin, we determined the aggregation size to be used for subsequent experiments. We then present performance results for each of our techniques. We conclude with a more detailed analysis that lets us gain some additional insight into how our protocols operate.

6.1 Maximum Aggregation Size

For our experiments, we needed to set a maximum aggregation size to be used in our further validation. We determined this size by considering throughput as a function of maximum frame size at various rates. For higher data rates the fixed overhead tends to have a

Table 2: 2-Hop UDP Throughput

Data Rate	No Aggregation	Unicast Aggregation	Difference
0.65 Mbps	0.253 Mbps	0.273 Mbps	7.9%
1.3 Mbps	0.430 Mbps	0.481 Mbps	11.9%

greater impact on the throughput, so we expected the improvement to be more significant for such rates.

To determine the maximum aggregation size, we used our UDP application on a 1-hop network. Setting the data interval (the time between transmissions) to 0.1 sec created enough queueing that aggregation becomes effective and noticeable. Figure 7 shows the throughput as a function of aggregation size for several rates. The X-axis shows the maximum aggregation size in Kbytes and the Y-axis shows the throughput in Mbps.

We observe that the throughput increases up to a threshold value and then rapidly falls off to 0. For higher data rates the throughput increase is steeper than for low data rates, which reflects the expected increased benefit of aggregation for higher rates.

The threshold value increases as the data rate increases. However, if we consider physical samples (the PHY uses "samples" with a variable number of bits to actually transmit data) instead of bytes, we observed a fixed threshold at about 120 Ksamples. This is consistent in our indoor environment. However, we will investigate the relationship between the maximum size and the channel condition. For the 0.65 Mbps rate, using BPSK and a $\frac{1}{2}$ coding rate, 120 Ksamples is 5 KB. For the 1.3 Mbps rate, using QPSK and a $\frac{1}{2}$ coding rate, 120 Ksamples is 11 KB. For the 1.95 Mbps rate, using QPSK and a $\frac{3}{4}$ coding rate, 120 Ksamples is 15 KB. These values match the thresholds shown in Figure 7. When we further increased the number of samples, subframes transmitted later in the process had uncorrectable errors and fail the CRC check resulting in the whole frame being discarded. We believe that this is because for longer frames, changes in the channel cause the channel estimates being used to become out of date, but we are investigating further to verify this theory. This phenomena represents a good example of where a partial/block ACK scheme might prove useful. In addition, we leave the possibility of changing the aggregation size as a function of rate to future work.

Based on these results, we chose 5 KB as the maximum aggregation size for all of the experiments below. This size allows our frame aggregation scheme to use all of the data rates provided by the PHY.

6.2 Unicast Aggregation

We designed this experiment to study the impact of unicast aggregation. Because aggregation saves MAC and PHY overheads as well as transmissions, we expect that throughput will be enhanced and that amount of enhancement will increase as rate increases.

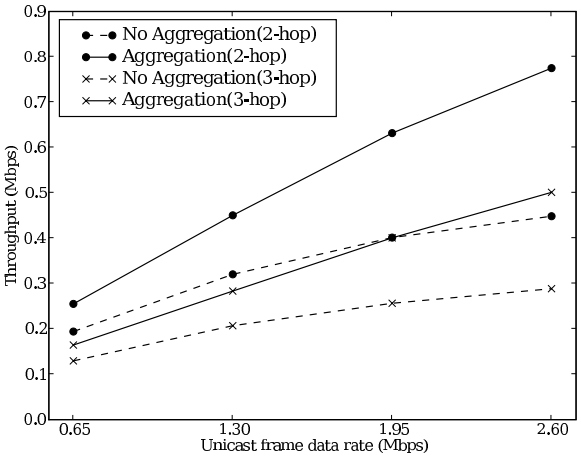

Figure 8: TCP Unicast Aggregation

For UDP traffic, we used our UDP application over a 2-hop network and fixed data rates of 0.65 Mbps and 1.3 Mbps. Table 2 presents throughput when the data interval is set to 3 seconds. We observed a significant improvement with aggregation and that, as expected, this improvement increases with rate.

For TCP traffic, we used a one-way file transfer over both 2- and 3-hop linear topologies. Figure 8 shows the experimental results for TCP as we varied the data rate. The X-axis shows the unicast data rate in Mbps and the Y-axis shows the end-to-end throughput in Mbps. Figure 8 demonstrates that throughput is improved for both 2-hop and 3-hop networks when aggregation is applied. Again, as expected, the improvement increases as data rate increases.

6.3 Broadcast Aggregation

We designed this experiment to assess the impact of broadcast aggregation in the presence of flooding. By combining the flooding frames with unicast frames, we expect that the impact of flooding on performance will be greatly reduced. For these experiments both unicast and broadcast aggregation was enabled.

We used our UDP application over a 2-hop network with a data interval of 3 seconds, and fixed data rates of 0.65 Mbps and 1.3 Mbps. To simulate flooding, each node generated broadcast frames at a fixed rate, which we varied during the experiments. Figure 9 shows the result of aggregation and no aggregation as a function of the flooding interval. The X-axis shows the interval of flooding frames in seconds and the Y-axis shows the end-to-end throughput in Mbps.

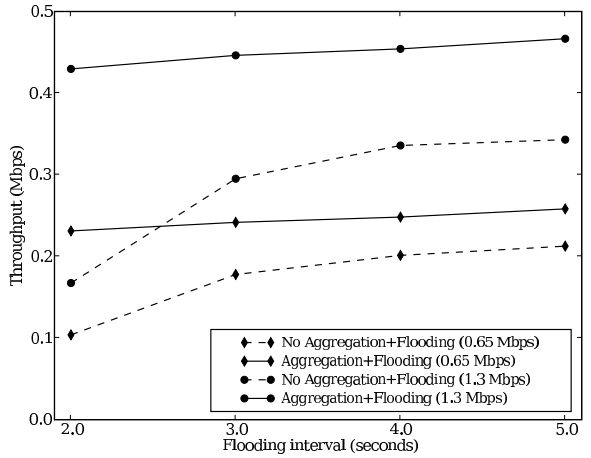

Figure 9: 2-Hop UDP Flooding

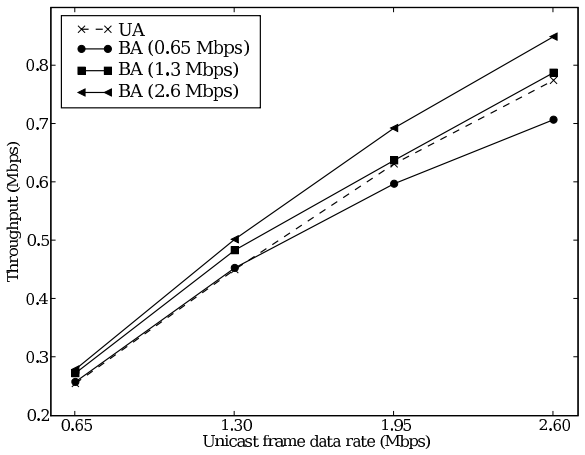

Figure 10: TCP ACK Aggregation with a Fixed Broadcast Rate

Our results show that for both data rates the performance gap between aggregation and no aggregation increases as the flooding interval decreases. A small flooding interval has a greater impact on the throughput when no aggregation is used, indicating that aggregation effectively reduces the overhead of flooding. We also observed the expected trend with increasing rate. If we compare this result with the unicast aggregation result when there is no flooding (Table 2), we find that for 0.65 Mbps with flooding at the 5 sec interval the throughput is 0.26 Mbps while without flooding it is 0.27 Mbps, while for 1.3 Mbps flooding has a throughput of 0.47 Mbps and without 0.48 Mbps. This degradation comes from two sources: first, only the throughput of the UDP application is considered; and second, as the flooding interval decreases, the number of aggregated frames containing only flooding frames increases.

6.4 TCP ACK Aggregation

We designed these experiments to study the effect of treating TCP ACKs as broadcast frames, thus enabling bi-directional aggregation. We study the impact of bi-directional aggregation in a variety of configurations. We start by measuring the throughput as a function of the unicast frame data rate, when the rate used by the broadcast subframes is fixed, followed by a measurement when the broadcast and unicast subframes use the same rate. We extend our experiments to 3-hop linear and star topologies to study the impact of hop count and network congestion. To study the impact of queueing delay and enhancements that might be gained by delaying frames to create greater opportunities for aggregation, we experimented with a version of our system that waited until it could aggregate 3 frames before transmission. To separate the impact of backward and forward aggregation, we performed experiments for TCP ACK aggregation while disabling forward aggregation. Finally, we present some detailed analysis that lends greater insight into behavior of our system.

All experiments in this Subsection use a one-way file transfer to generate TCP traffic. We use the abbreviation BA to denote broadcast aggregation results, UA for only unicast aggregation, and NA for no aggregation.

6.4.1 2-hop Networks

We designed these experiments to compare TCP performance between BA and UA over a 2-hop network by studying throughput as a function of PHY data rate. In the first experiment, we varied the rate of the unicast part of the frame, but fixed the rate of the broadcast part of the frame. In the second experiment, the broadcast and unicast subframes have the same rate.

Figure 10 shows the results for BA when using the fixed data rates 0.65 Mbps, 1.3 Mbps, and 2.6 Mbps for the broadcast TCP ACKs, and for UA. The X-axis shows the unicast data rate measured in Mbps and the Y-axis shows the end-to-end throughput in Mbps. The value in parenthesis is the fixed rate used for the broadcast subframes.

When the MAC broadcasts TCP ACKs at 0.65 Mbps, we saw that BA shows some improvement over UA at the unicast rate 0.65 Mbps, but that as the unicast rate increases, the throughput of BA falls off. As the unicast rate goes up, the time spent transmitting the broadcast ACK at 0.65 Mbps increasingly dominates, causing this effect. When TCP ACKs are broadcast at 1.3 Mbps, we observe that BA outperforms UA up to the unicast rate of 1.3 Mbps, and afterwards BA achieves performance similar to UA. BA using 1.3 Mbps becomes more effective because for high data rates the impact of aggregation becomes more significant. When TCP

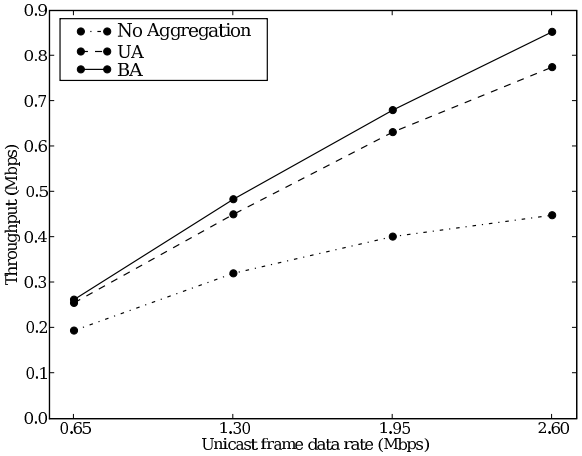

Figure 11: 2-hop TCP ACK aggregation

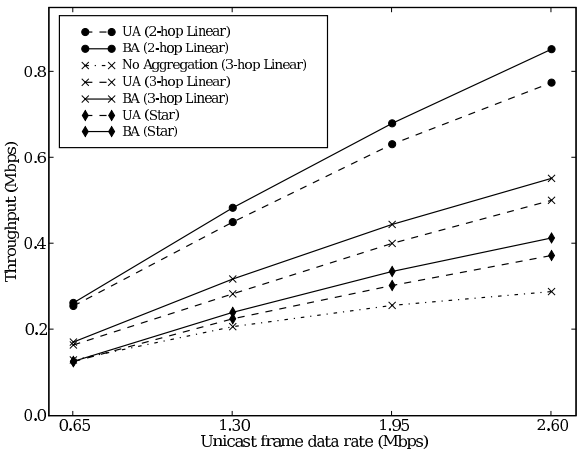

Figure 12: TCP over more complex topologies

ACKs are broadcast at 2.6 Mbps, we observe that BA always outperforms UA over the range of rates used in our experiments, for the same reasons we have already mentioned.

Figure 11 shows similar results when the broadcast TCP ACKs use the same data rate as the unicast subframes. In this case BA always outperforms UA. Comparing BA with UA, the maximum throughput performance gap is 10%. (Also, improvements exist at 0.65 Mbps, but are hard to see at the scale used in the graph.) This is because BA can aggregate more subframes at the relay nodes both reducing header overhead and the number of transmissions. We also include the no aggregation results to show that both BA and UA provide significant improvements over not doing any aggregation, as expected. Because it provided consistently better performance, we use this version with broadcast and unicast frames at the same rate for all subsequent experiments.

6.4.2 3-hop Linear and Star Networks

We designed these experiments to study the performance of our system with more complex topologies, such as when more relay nodes become involved or when the network becomes congested. We expected that BA would show an enhancement over UA due to increased hop count and that network congestion would increase the ratio of aggregation. In a star topology, we expected more congestion because the central node would be a bottleneck for the TCP streams. More congestion results in longer queues which provides more chances for aggregation. In a 3-hop linear topology, we expected that adding a relay node would increase the number of nodes involved in aggregation of TCP data and ACKs, improving the aggregation ratio.

We used two topologies: A 3-hop linear topology (Figure 5) having more hops and a star topology (Figure 6) having more congestion. The star topology creates two TCP sessions, with each session over 2 hops. For the star topology, we measured the worst-case throughput over the paths, i.e., the throughput for the slowest session. For this experiment, broadcast ACKs were transmitted at the unicast rate. Figure 12 shows the throughput as a function of data rate. The X-axis shows data rate measured in Mbps and the Y-axis shows the end-to-end throughput in Mbps.

As expected, BA shows more improvement in both scenarios. For the 3-hop linear topology, the maximum throughput gap between BA and UA is 12.2% as compared to 10% for 2-hop. As more relay nodes become involved in bi-directional aggregation, the overall aggregation ratio increases. For the star topology, the maximum throughput gap is 11%, because network congestion leads to longer queues. This allows BA to have more aggregation opportunities than UA, because UA only supports aggregating frames being transmitted to the same receiver. Note, we also show the no aggregation result for 3-hop, which has the expected performance. In addition, we expect that, for star topologies, the result of no aggregation will be worse than that of UA and BA. This is clear because the no aggregation does not support any methods reducing the cost of TCP ACKs.

6.4.3 Delayed Aggregation

We designed this experiment to study the impact of the queueing delay on our aggregation approach. We expect that longer queues will result in more opportunities for aggregation.

We created a delayed version of BA, delayed BA (DBA), which forces relay nodes to pause transmission until they have 3 frames in their queues. We choose 3 frames

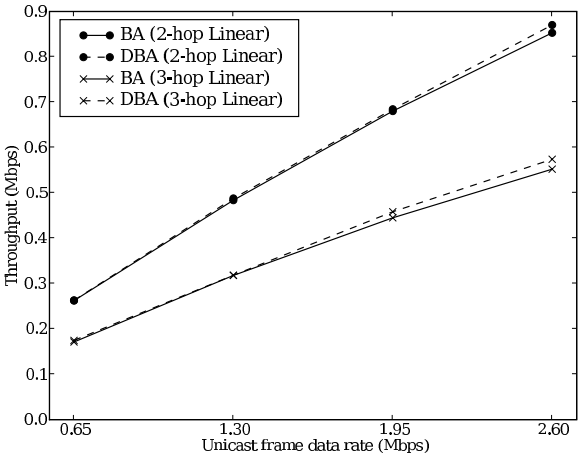

Figure 13: TCP for Delayed BA

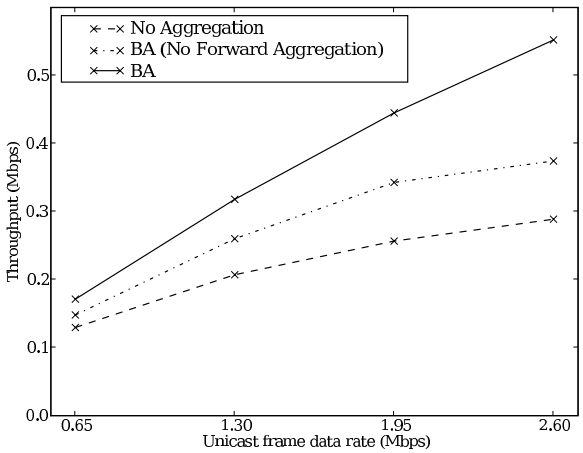

Figure 14: TCP without forward aggregation

because that is the maximum number of TCP data frames that can be aggregated, given the parameters of our experiments. Figure 13 shows the performance comparison between BA and DBA over both of 2-hop and 3-hop linear topologies as a function of rate. The X-axis shows the unicast data rate in Mbps and the Y-axis shows the end-to-end throughput in Mbps.

We observed that the performance of BA and DBA is similar at the unicast rates of 0.65 Mbps and 1.3 Mbps and at higher rates DBA outperforms BA slightly. The maximum gap is 2% and 4% for 2-hop and 3-hop networks respectively. This improvement is smaller than we expected and we are investigating further.

6.4.4 *Forward vs. Backward Aggregation*

We defined "forward aggregation" to mean the aggregation of packets going in the same direction, and "backward aggregation" to mean the aggregation of packets flowing in the opposite direction, such as TCP data and ACKs. We designed this experiment to gain some insight about where the benefits from backward aggregation occur. We do this by disabling forward aggregation so that the benefit from aggregation comes purely from combining TCP data and ACKs into one transmission.

For this experiment, the 3-hop linear topology was used. Figure 14 compares the performance of no aggregation, BA, and BA without forward aggregation. The X-axis shows the data rate in Mbps and the Y-axis shows the end-to-end throughput in Mbps.

The results show that the performance gap between BA and BA with disabled forward aggregation becomes bigger as the unicast data rate increases. This means that backward and forward aggregation affect the throughput equally when low data rates are used. As the data rate increases, forward aggregation has a greater impact on the throughput. This probably reflects a limit in the level of bi-directional aggregation possible in our benchmarks.

6.4.5 *Detailed Analysis*

Although the results shown thus far show the basic performance characteristics of our system, a more detailed analysis can give us more insight about the performance of DBA, BA, UA, and NA. We start by focusing on the impact of bi-directional aggregation in a 2-hop network and then we extend our analysis to star and 3-hop networks to gain insight about the impact of the topology on performance.

2-hop Linear Topology.

Table 3 shows the average frame size (in bytes), the number of transmissions (TX) (as a percentage of the no aggregation number), and the size overhead (as a percentage of the MAC and PHY header size compared to total size) of NA, UA, BA and DBA in a 2-hop network. The table shows that the average frame size increases dramatically when we introduce aggregation and somewhat more modestly when broadcast and then delayed aggregation are introduced. A maximum TCP data frame is 1464B and a TCP ACK is 160B the average of which is 812B. This is close to the average NA size, suggesting that some TCP data segments are smaller than maximum size. On the other hand the average size for UA is 2662B which divided by 812B is 3.3. For the parameters we are using, 3 maximum size TCP data segments will fit in one frame, which suggests that for UA we are typically fully aggregating data frames and perhaps sometime sending as many as 4 ACKs at once. Thus TCP effectively expands its window to take advantage of aggregation. BA and DBA both achieve slightly better aggregation as expected. The results for transmissions bear this analysis out, since aggregating 3 data

Table 3: 2-hop Relay Node Detail

	NA	UA	BA	DBA
Frame Size	765B	2662B	2727B	3477B
Total TXs	100%	33.7%	26.7%	21.1%
Size overhead	15.1%	6.83%	6.55%	5.8%

or 3 ACK frames in each transmission would result in one-third the number of transmissions, exactly as seen. Again, the greater aggregation of BA and DBA result in somewhat fewer transmissions compared to UA. Finally, the differences in header overheads reflect these same trends as we would expect.

Table 4 presents the average percentage time overhead as a function of the data rate. The overhead includes the transmission time for MAC and PHY headers, the transmission time for control frames, backoff, DCF interframe space (DIFS), and short interframe space (SIFS). This table shows very clearly how at higher data rates overhead begins to dominate. In the no aggregation case it goes from 22.4% to 52.1%. It also shows that aggregation is very effective in reducing this overhead.

Star Topology vs. 2-hop Linear Topology.
Table 5 shows the average frame size in both 2-hop and star topologies. The frame size is almost constant for UA, reflecting the fact that only frames with the same destination can be aggregated and thus there are no greater possibilities for aggregation in the star topology than in the 2-hop one. For BA there is a substantial increase for the star because TCP ACKs destined for node 3 and 4 can be aggregated together and with TCP data frames destined for node 1. Table 6, shows the size overheads for the two topologies. It shows a similar trend as frame size. Table 7 shows the number of transmissions relative to the no aggregation case for the two topologies. It shows similar trends as the previous two results, although UA shows fewer transmissions for the star topology. This is because UA suffers from queue overflow, which results in sending fewer packets for a given file size. On the other hand, for NA, we multiply the total number of transmissions by two because we do not have the NA results for the star topology. These decrease the transmission percentage for UA.

Table 4: 2-Hop Relay Node Time Overhead

Data Rate	NA	UA	BA	DBA
0.65	22.4%	6.7%	5.8%	5.2%
1.3	34.9%	14.3%	11.4%	10.3%
1.95	44.4%	19.3%	15.5%	14.3%
2.6	52.1%	24.8%	19.9%	17.7%

Table 5: Relay Node Frame Size

	2-hop	Star
UA	2662B	2651B
BA	2727B	3432B

Table 6: Relay Node Size Overhead

	2-hop	Star
UA	6.83%	6.83%
BA	6.55%	5.93%

3-hop Linear Topology vs. 2-hop Linear Topology.
Table 8 shows the frame size of the TCP server, relay node(s), and TCP client in both 2-hop and 3-hop networks. For both BA and UA, the frame sizes at the TCP server and client imply that the server transmits an aggregated frame containing two (2928B) or three (4392B) subframes and, as a response, the client sends two (320B) or three (480B) ACKs back to the server.

The data shows that, at the relay nodes, the average frame size in a 2-hop network is bigger than that in a 3-hop network. This is because the TCP data and TCP ACK transmission rates decrease for 3-hops. The reduced transmission rates affect the average sizes at the TCP server and client. The average frame sizes of BA at the TCP server and client are 3488B and 447B in a 2-hop network respectively. Meanwhile, the average frame size decreases to 3313B and 430B in a 3-hop network. These reduced sizes make the average frame size at the relay nodes decrease. However, it is useful to compare the sizes at the relay nodes. For 2 hops, the difference between UA and BA is 65B, while for 3 hops at relay1 it is 154B and at relay2 it is 446B. Thus we see a significant increase in the amount of aggregation as the number of hops increases.

7. CONCLUSIONS AND FUTURE WORK

We presented a design for unicast aggregation, broadcast aggregation, and for treating TCP ACKs as broadcasts so as to enable them to be aggregated. We used our wireless networking prototype, Hydra, to implement our aggregation schemes and to perform a series of experiments to validate our design. The performance results are promising for all approaches.

Possible future work falls broadly into two categories,

Table 7: Relay Node Transmission Percentages

	2-hop	Star
UA	33.7%	30.7%
BA	26.7%	22.5%

Table 8: Frame Size at all Nodes for 2-hop and 3-hop Networks

	Server (2^+)	Relay (2)	Client (2)	Server (3)	Relay1 (3)	Relay2 (3)	Client (3)
UA	3897	2662	463	3451	2384	2224	443
BA	3488	2727	447	3313	2538	2670	430

$^+$ indicates the number of hops.

experimental work and protocol development and validation. Experimentally, we plan to perform further experiments when both TCP and standard broadcast frames are generated. In addition, to verify the impact of treating TCP ACKs as broadcasts on performance, we will extend our experiments to the large-scale networks using simulation.

In terms of protocols, we plan to extend our system to make better use of frame aggregation by implementing a block ACK scheme. As an extension to our existing rate adaptation protocol, we plan to design and implement a rate-adaptive frame aggregation scheme. By adapting rates for broadcasts and unicasts, we anticipate further performance improvements. In addition, we will implement a delayed ACK scheme that allows the MAC to aggregate unicast subframes which have different destination addresses.

8. REFERENCES

[1] Enhanced Wireless Consortium. *HT MAC Specification v1.24*, 2006.

[2] S. Floyd and T. Henderson. The NewReno Modification to TCP's Fast Recovery Algorithm. *IETF RFC 2582*, Apr. 1999.

[3] GNU Radio: Universal Software Radio Peripheral. `http://www.gnuradio.org/trac/wiki/USRP`.

[4] GNU Software Radio. `http://www.gnu.org/software/gnuradio/`.

[5] G. Holland, N. H. Vaidya, and P. Bahl. A Rate-Adaptive MAC protocol for Multi-Hop Wireless Networks. In *Proceedings of ACM MOBICOM*, July 2001.

[6] Hydra – A Wireless Multihop Testbed. `http://hydra.ece.utexas.edu/`.

[7] IEEE 802.11 Working Group, Piscataway, NJ. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, 1997.

[8] IEEE 802.11n Working Group. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification-Draft 2.0: Enhancements for Higher Throughput, Part 11 Standard ed.*, 2007.

[9] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Proceedings of ACM SIGCOMM*, Aug. 1996.

[10] A. Kamerman and L. Monteban. WaveLAN II: A High-Performance Wireless LAN for the Unlicensed Band. *Bell Labs Technical Journal*, 2(3):118–133, 1997.

[11] S. Kim, S. Choi, Y. Kim, and K. Jang. MCCA: A High-Throughput MAC Strategy for Next-Generation WLANs. *IEEE Wireless Communications*, 15(1):32–39, Feb 2008.

[12] Y. Kim, S. Choi, K. Jang, and H. Hwang. Throughput Enhancement of IEEE 802.11 WLAN via Frame Aggregation. In *Proceedings of IEEE VTC*, Sept. 2004.

[13] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click Modular Router. *ACM Transactions on Computer Systems*, 18(3):263–297, 2000.

[14] C. Parsa and J. J. Garcia-Luna-Aceves. Improving TCP Performance over Wireless Networks at the Link Layer. *Mobile Networks and Applications*, 5(1):57–71, Mar. 2000.

[15] C. Perkins, E. Royer, and S. Das. Ad-hoc On-demand Distance Vector (AODV) Routing. In *Proceedings of IEEE WMCSA*, Feb. 1999.

[16] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. Opportunistic Media Access for Multirate Ad hoc Networks. In *Proceedings of ACM MOBICOM*, Sept. 2002.

[17] L. Scalia, F. Soldo, and M. Gerla. PiggyCode: a MAC Layer Network Coding Scheme to improve TCP Performance over Wireless Networks. In *Proceedings of IEEE GLOBECOM*, Nov. 2007.

[18] D. Skordoulis, Q. Ni, H.-H. Chen, A. P. Stephens, C. Liu, and A. Jamalipour. IEEE 802.11n MAC Frame Aggregation Mechanisms for Next-Generation High-Throughput WLANs. *IEEE Wireless Communications*, 15(1):40–47, Feb 2008.

[19] J. Tourrilhes. PiggyData: Reducing CSMA/CA Collisions for Multimedia and TCP Connections. In *Proceedings of IEEE VTC*, Sept. 1999.

[20] Y. Xiao. Concatenation and Piggyback Mechanisms for the IEEE 802.11 MAC. In *Proceedings of IEEE WCNC*, Mar. 2004.

[21] Y. Xiao. IEEE 802.11n: Enhancements for Higher Throughput in Wireless LANs. *IEEE Wireless Communications*, 12(6):82–91, Dec. 2005.