An Empirical Evaluation of Bufferbloat in IEEE 802.11n Wireless Networks

Ahmad Showail, Kamran Jamshaid, and Basem Shihada
Computer, Electrical, and Mathematical Sciences and Engineering (CEMSE) Division
King Abdullah University of Science and Technology
Thuwal, Saudi Arabia
Email: {ahmad.showail, kamran.jamshaid, basem.shihada}@kaust.edu.sa

***Abstract*—In this paper, we analyze the impact of large, persistently-full buffers ('bufferbloat') on various network dynamics in IEEE 802.11n wireless networks. The bufferbloat problem has mostly been studied in the context of wired networks. We study the impact of bufferbloat on a variety of wireless network topologies, including wireless LAN (WLAN) and multi-hop wireless networks, with varying wireless link rates. Our results show that a single FTP transfer between two Linux wireless hosts can saturate the buffers in the network stack, leading to RTT delays exceeding 4.5 s in multi-hop configurations. We show that Aggregate MAC Protocol Data Unit (A-MPDU) MAC-layer frame aggregation can be used to reduce RTT delays while simultaneously increasing network throughput. However, additional measures may still be required to meet the constraints of real-time flows (such as VoIP). Our experiments show that large buffers can deteriorate the fairness in rate allocation in parking lot based multi-hop networks.**

***Index Terms*—Bufferbloat, IEEE 802.11n, Frame Aggregation, A-MPDU, TCP**

I. Introduction

The impact of 'persistently-full', large buffers on network performance have been known for many years. These buffers build up at network bottlenecks along the routing path of a flow. Recently, the term 'bufferbloat' [2] has been used to describe the performance impact when these large buffers are used with simplistic FIFO queue management with drop tail packet scheduling. Most of the existing work on buffer sizing has been studied in the context of core Internet routers with a large number of flows (*e.g.,* [5], [7], among others). With the network core increasingly being over-provisioned and popularity of wireless hand-held devices, the bottlenecks often lie in the access network. For example, many enterprise users accessing server applications via their corporate wireless LAN (WLAN) are bottlenecked by the wireless link capacity. Similarly, users accessing the Internet via a metropolitan area Wi-Fi Mesh Network (WMN) or a 3G/4G cellular network, are likely to be bottlenecked by the slow wireless link capacity. Thus, it is important to study the impact of bufferbloat on wireless network performance.

In this paper we study the impact of bufferbloat on high-speed IEEE 802.11n wireless networks. The wireless environment brings new challenges to our understanding of buffer sizing requirements [11]. In particular, wireless networks have time-varying link rates, variable packet inter-service time, as well as experience interference from other devices sharing the same frequency spectrum. In addition, various enhancements for improving efficiency, such as frame aggregation, also impact packet scheduling dynamics that need to be considered while designing buffer sizing mechanisms.

The MAC layer in IEEE 802.11n standard introduces two types of frame aggregation techniques [9]: Aggregate MAC Protocol Service Data Unit (A-MSDU) and Aggregate MAC Protocol Data Unit (A-MPDU). A-MSDU aggregates multiple MAC Service Data Units (MSDUs) destined to a common receiver into a single MAC Protocol Data Unit (MPDU). It then appends a single MPDU header, as well as a Frame Check Sequence (FCS) trailer. The maximum A-MSDU frame size that a receiver can process is advertised in the HT Capabilities Information field, and is either 3,839 bytes or 7,935 bytes. A-MPDU consists of multiple MPDUs aggregated into a frame. Each MPDU has its own FCS field. This allows a receiver to request a retransmission of corrupted MPDUs by transmitting a Block Ack frame containing a bitmap to identify the status of individual MPDUs. A-MPDU is limited in size to 65,535 bytes (bound by the 16-bit length field in the HT-SIG headers) and can carry a maximum of 64 sub-frames (limited by the bitmap in the Block ACK frame). The actual A-MPDU size used for communication may further be limited by the HT receiver, as advertised in its HT Capabilities element. The aggregation logic is not specified in the standard, and thus it varies between vendors based on their implementation. Ideally, these algorithms need to balance the requirements between making efficient use of channel resources using large frames aggregates when the channel is good, while attempting to minimize queueing delays and processing packets quickly.

Our main contributions in this paper are as follows: we characterize the impact of bufferbloat on a 802.11n testbed using both single hop and multi-hop topologies. We show that a suitably designed A-MPDU aggregation scheduler can substantially reduce delays while simultaneously increasing throughput. Our experiments span a range of wireless link rate configurations and our observations can be used to design optimal buffer sizing mechanisms for wireless networks.

The remainder of this paper is organized as follows. We first summarize the background and related work. Next, we describe our experimental setup and the A-MPDU aggregation scheduler used in our experiments. Our performance results are described in Sect. V. Finally, we conclude in Sec. VI.

II. Related Work

Buffer sizing has been extensively studied for wired networks. A widely used rule-of-thumb is to have buffers larger than the bandwidth-delay product (BDP) of the network [17], *i.e.*, $B \geq RTT \times C$, where C is the *bottleneck* link capacity along the path, and RTT is the effective round-trip propagation delay through the bottleneck link. The buffer size B reflects the number of packets queued at a bottleneck link to keep it fully utilized while a TCP source recovers from a loss-induced window reduction. This rule-of-thumb holds for a single TCP flow in a wired network. When a large number of flows (say, N) share a bottleneck router, the window size processes quickly synchronize in lockstep. As a result, the aggregate window size is still a sawtooth pattern and the BDP guideline for sizing the bottleneck buffer still holds. However, when the flows are desynchronized and the window processes are independent, the buffer size B can be reduced to $B = RTT \times C/\sqrt{N}$ while still achieving near 100% utilization [5]. Enachescu *et al.* [7] suggest that B can further be reduced to $O(log\ W)$, where W is the window size of each flow, resulting in buffer sizes of only $10-20$ packets while achieving $85-90\%$ of link utilization.

There has been a limited amount of work on buffer sizing for wireless networks. For single-hop 802.11 WLANs, large Access Point (AP) buffers can improve fairness between upstream and downstream TCP flows [6]; a large buffer increases the queueing delay for TCP ACKs being transmitted back to the upstream wireless source, in effect rate limiting the upstream flows. This scheme, however, disturbs the tight feedback loop necessary for optimal operation of TCP. The AP buffer can be sized dynamically to strike a balance between channel utilization and delay requirements of various flows [14].

For multihop wireless networks, network congestion can be detected by monitoring queue sizes. Xu *et al.* [18] have extended this notion to a distributed 'neighborhood queue'. Their NRED algorithm improved flow rate fairness by probabilistically dropping packets when the cumulative size of this distributed queue exceeds a threshold. Jamshaid *et al.* [11] have proposed a distributed buffer sizing scheme using a similar notion of a 'neighborhood buffer' spanning a set of interfering nodes that constitute the network bottleneck. Using 802.11b networks, they found that small buffers (as low as 1-4 packets) at each node are sufficient to provide high network throughput without incurring large queueing delays. However, it is unclear if similar buffer sizes would also be useful in 802.11n networks with much higher link rates.

The impact of packet aggregation in 802.11n has been discussed in prior work [8], [16]. The authors show that frame aggregation improves network throughput, though its impact on end-to-end delays, and in particular, its relation to buffer sizes is not considered.

Recent work has also uncovered overbuffering in 3G/4G mobile networks [13]. The experimental results showed RTT latency exceeding 1 s across the 3G networks of four major US carriers. As a counter-measure, some mobile phones based on the Android platform use a preset, small maximum receive window size advertised by the receiver to limit the growth of the TCP congestion window. However, this small size leads to a suboptimal throughput in networks with large BDP (such as high speed, long distance networks). Similarly, for networks with low BDP, this preset value is too large and results in excessive RTT delays.

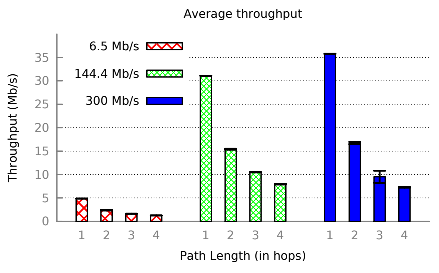
III. Experimental Setup

Our testbed consists of small form-factor Shuttle computers [3] with an Intel E7500 Core 2 Duo processor, 1 GB of DRAM, and a TP-Link WDN4800 (Atheros AR9380) 802.11 a/b/g/n wireless card. The chipset supports three spatial MIMO streams for a maximum wireless link rate of 450 Mb/s. Our network uses the 5 GHz U-NII (Unlicensed National Information Infrastructure) radio band. This does not interfere with the production WLAN on our campus which uses the 2.4 GHz spectrum. We run a custom 3.3 Linux kernel with Web10g [4] instrumentation to monitor the state characteristics of our TCP streams. It uses an efficient Netlink-based kernel Application Binary Interface to make the TCP statistics available in userspace. We use ath9k drivers [1] to configure the wireless interfaces. We disabled wireless link rate adaptation algorithms, and set link rates manually. These link rates are varied, as specified alongside each experiment. We use iperf and netperf to generate traffic. Our Linux distribution uses TCP Cubic. Window scaling [10] is enabled, as per the default configuration on all recent Linux kernels. This enables TCP to support a receive window size greater than 64 kB. Our wireless configuration parameters are summarized in Table I.

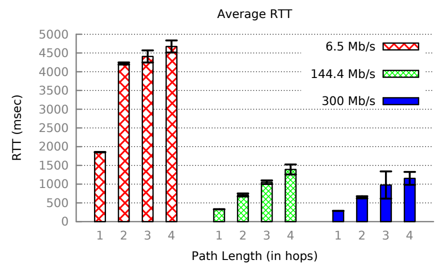
Parameter	Value
Link rates	6.5 Mb/s, 144.4 Mb/s, 300 Mb/s
MAC protocol	IEEE 802.11n
Traffic source	iperf, netperf
Packet size	1500 Bytes
txqueue size	1000 packets (Default size)
TCP Flavor	Cubic with window scaling
Test duration	200 s
Routing	Fixed path routing

TABLE I: Experimental setup

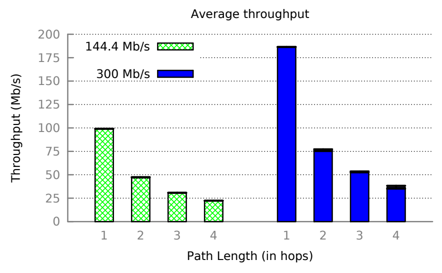
The ath9k driver supports A-MPDU transmission (both A-MSDU and A-MPDU reception is mandatory per IEEE 802.11 standard specifications). The maximum aggregate size achieveable in practice is limited to a frame duration of 4 ms to comply with regulatory requirements for operation in the 5 GHz U-NII band. Thus the actual density of MPDUs in an A-MPDU is partly dependent on the wireless link rate which determines the frame transmit duration. We conduct experiments both with and without A-MPDU aggregation to isolate the impact of aggregation on wireless network performance. Each experiment for a given link rate and network topology is performed 5 times. The throughput and RTT results are then averaged across these runs.



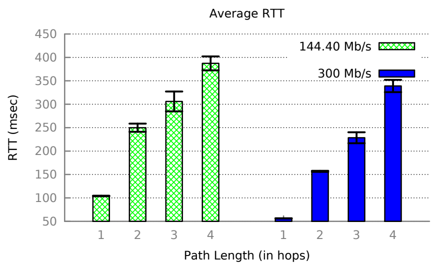
(a) Throughput without aggregation



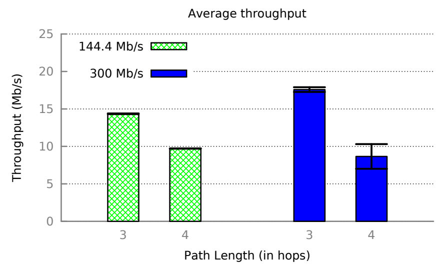
(b) Delay without aggregation



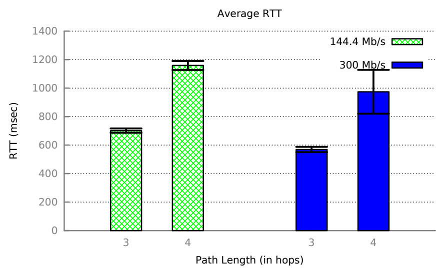
(c) Throughput with aggregation



(d) Delay with aggregation



(e) Throughput with partial aggregation



(f) Delay with partial aggregation

Fig. 1: Flow throughput and RTT for links with and without aggregation and with partial aggregation

IV. A-MPDU Aggregation Algorithm

The IEEE 802.11n standard does not specify the packet scheduler for assembling A-MPDUs for transmission. A naïve packet scheduler that always waits to assemble a maximum sized A-MPDU frame for transmission can potentially further deteriorate the delay performance. The A-MPDU aggregation logic used in our setup is listed in Algo. 1. This algorithm balances delay and throughput trade-off by preferring timeliness over capacity; instead of waiting to assemble maximal allowable A-MPDU aggregates which may maximize throughput, we aggregate as many MPDUs as available at that time in the buffer subject to the regulatory and receiver constraints. Thus, while we may not use optimal A-MPDU sizes, the fact that we never spend time waiting for new frames to arrive from higher layers can result in improving end-to-end delays while increasing throughput. We analyze the performance of this algorithm in Sect. V.

```
Input: Number of frames in buffer ($Q$), Regulatory A-MPDU size limit
       ($\delta_1$), Receiver A-MPDU size limit ($\delta_2$), Number of frames in
       this A-MPDU ($n$)
Output: Assemble A-MPDU for transmission
n = 0, A-MPDU = 0;
while $Q \neq 0$ do
    // Check for regulatory or receiver limit;
    if ($n > \delta_1$ or $n > \delta_2$) then
        break;
    end
    Add padding (if necesary) to A-MPDU to align frame boundry;
    Link this frame to the aggregate A-MPDU;
    Q- -;  // Decrement buffer count by 1;
    n++;  // Increment frame count by 1;
end
Deliver assembled A-MPDU to driver transmit function;
```

Algorithm 1: Algorithm for A-MPDU aggregation logic

V. Experimental Analysis

We conducted a number of experiments to understand the impact of bufferbloat on wireless network performance. We used three link rates for our experiments: 6.5 Mb/s (Modulation and Coding Scheme (MCS) index 0, 20 MHz channel bandwidth, 800 ns Guard Interval (GI)), 144.4 Mb/s (MCS 15, 20 MHz channel, 400 ns GI), and 300 Mb/s (MCS 15, 40 MHz channel, 400 ns GI).

A. Single-flow topologies

We first document the performance analysis of a single TCP flow. We disable transmit A-MPDU aggregation for our first set of results. We then vary the path length of the flow from a single-hop network to a 4-hop chain topology. A given experiment uses a uniform wireless link rate between adjoining nodes. This rate is varied from 6.5 Mb/s, 144.4 Mb/s, and 300 Mb/s in different experiments. Our results are shown in Figs. 1a and 1b. The error bars represent the standard deviation.

As expected, throughput increases with the link rate, and shows a decreasing trend with the hop-count. The throughput drops by $\frac{1}{2}$, $\frac{1}{3}$, and $\frac{1}{4}$, for 2, 3, and 4-hop networks, suggesting sparse spatial reuse in our topologies. The difference in throughput between 144.4 Mb/s links and 300 Mb/s links is small, and almost within error bounds of our measurements for 3 and 4-hop topologies. Our RTT delay measurements clearly show the impact of bufferbloat. The average 1-hop RTT delay measurements are 1853 ms, 328 ms, and 286 ms for 6.5 Mb/s, 144.4 Mb/s, and 300 Mbp/s links. These delays show that a single file-transfer between two wireless nodes can saturate the device buffers, even at the 300 Mb/s link rate, as shown in

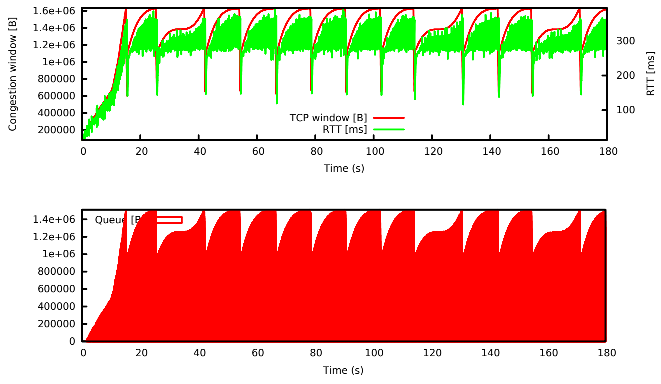


Fig. 2: TCP congestion window, RTT, and egress queue utilization for a 1-hop TCP flow over a 300 Mb/s wireless link

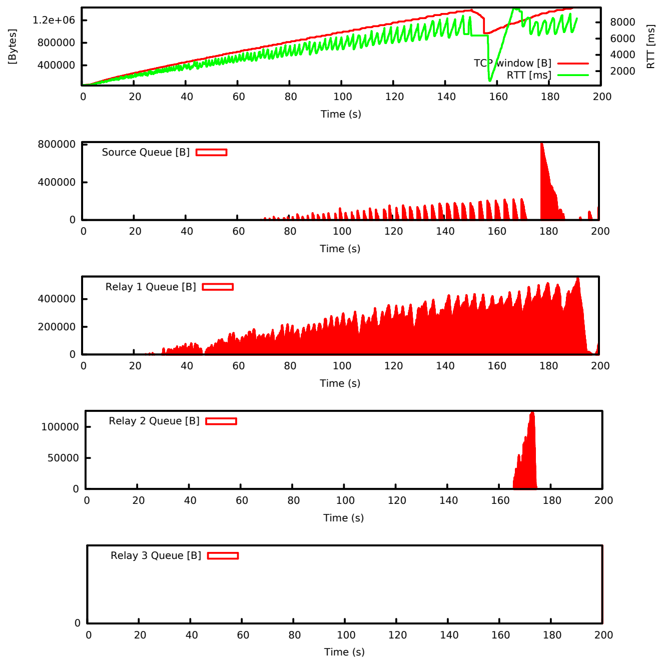


Fig. 3: TCP congestion window size, RTT, and txqueue size distribution for a TCP flow in a 4-hop chain topology with 6.5 Mb/s wireless links

Fig. 2. We observe that the buffer size grows up to its limit of a 1000 packets (each packet of 1500 bytes), before registering a queue drop and triggering TCP's congestion control algorithm.

In general, we observe that while RTT measurements increase with the hop count, they do not always exhibit a proportionate increase, *e.g.*, for 6.5 Mb/s links, a 3-hop and a 4-hop network shows an increase of only 4.5% and 10% over the 2-hop delays. This suggests that with slow speed links, TCP's feedback mechanism is unable to saturate all buffers along the multi-hop path. This is also validated by the buffer utilization plots shown in Fig. 3, where none of the buffers fills up to the capacity. In such networks, TCP's congestion control mechanisms are triggered by losses other than queue drops, such as wireless collisions. In contrast, topologies using 144 Mb/s or 300 Mb/s links show a more uniform, consistent increase in RTT with increasing hop-count.

Next, we enable A-MPDU transmit aggregation based on the scheduler described in Sect. IV. Our measurement results are shown in Figs. 1c and 1d. The ath9k release used in our experiments does not support A-MPDU aggregation at 6.5 Mb/s, as transmitting a large A-MPDU at this link rate may violate the 4 ms frame transmit duration regulatory requirement in 5 GHz band. Thus, we only show the results for 144.4 Mb/s and 300 Mb/s link rates. A-MPDU aggregation significantly increases the network throughput. For a 1-hop network, 144.4 Mb/s link shows a throughput improvement of $3\times$, while a 300 Mb/s link shows an improvement of $5\times$. For multi-hop networks using 144.4 Mb/s, the throughput rougly decreases by $\frac{1}{2}$, $\frac{1}{3}$, and $\frac{1}{4}$ for 2, 3, and 4-hop chain topologies. However, the throughput drop is higher when using the 300 Mb/s link rate, averaging across 60% for each additional hop. To investigate further, we measured the A-MPDU size (in terms of MPDUs per A-MPDU) at each hop along the path to the destination. The average A-MPDU size is shown in Fig. 4. We observe that for 1-hop networks, the A-MPDU size approaches the maximum limit of 32 MPDUs imposed by our device driver. However, the average A-MPDU size is smaller in multi-hop networks. This is because the packets are dispersed in queues at multiple nodes along the path to the destination, and thus a given node may not always have the maximum number of MPDUs ready to transmit together. In particular, 300 Mb/s links use a smaller A-MPDU size compared to 144 Mb/s links for the 2nd, 3rd, and 4th relay nodes in the multi-hop topologies, leading to higher than 50% drop in throughput over a 1-hop 300 Mb/s link.

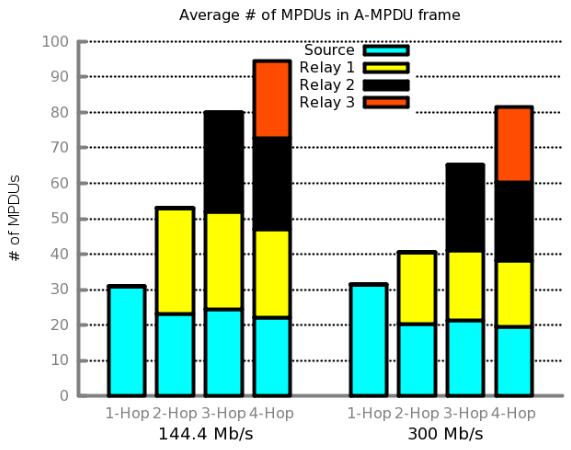


Fig. 4: Average A-MPDU size. For multi-hop networks, we report the A-MPDU size measured at each hop along the path to the destination. ath9k does not support Tx A-MPDU aggregation at 6.5 Mb/s link rate.

In addition to the increase in throughput, A-MPDU aggregation also considerably lowers the RTT across all topologies as shown in Fig. 1d, *e.g.,* the 4-hop RTT improves by over $3\times$ for both link rates. We observe that our A-MPDU scheduler does not always transmit a maximum sized A-MPDU, as illustrated in Fig. 4. Our measurements showed that many A-MPDUs were transmitted with a smaller size based on the number of frames available in the buffer at a given time. Transmitting multiple MPDUs together debloats the txqueue size, leading to smaller queueing delays and the subsequent reduction in RTT.

Finally, we conducted an additional set of single flow experiments for multi-hop topologies where only part of the nodes used transmit A-MPDU aggregation. Such networks are likely in heterogeneous environments with a mix of equipment from different vendors or support for backward compatibility with 802.11 a/b/g technologies. The source node and the first relay node had A-MPDU aggregation enabled, while all subsequent relay nodes had A-MPDU aggregation disabled. Our throughput and RTT measurement results are shown in Figs. 1e and 1f. We omit the 1-hop and 2-hop results, as those are similar to results in Figs. 1c and 1d.

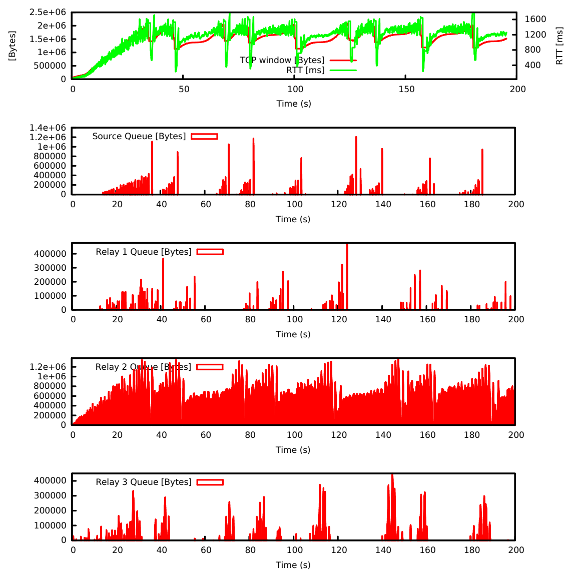


Fig. 5: TCP congestion window size, RTT, and txqueue size distribution for a TCP flow in a 4-hop chain topology with 144.4 Mb/s wireless links. The source and 1-hop relay node use Tx A-MPDU aggregation

We observe that throughput drops below the values observed when all links used transmit A-MPDU aggregation (Fig. 1c), though its still higher than values obtained when A-MPDU aggregation is disabled for all links (Fig. 1a). Similarly, our RTT measurements also fall in between the values obtained with these two cases. This suggests that even having some partial nodes using A-MPDU transmission in a network can increase the throughput and reduce delays. In such scenarios, the network is bottlenecked by the nodes that do not transmit using A-MPDU aggregation. We measured the queue utilizations across these topologies. One representative result is shown in Fig. 5 for a 4-hop chain topology using 144.4 Mb/s wireless links. Our analysis shows that sustained queues mostly build up at the first node that does not support A-MPDU transmit aggregation. For our 4-hop chain topology, this is the relay node 2-hops from the source. A buffer sizing strategy for reducing queueing delays would need to identify and manage similar set of bottlenecks in the network.

B. Multi-flow topologies

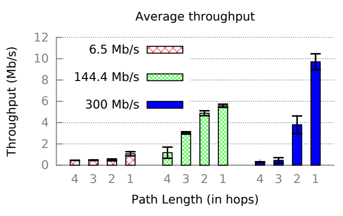
In this set of experiments, we characterize the impact of bufferbloat in wireless networks with multiple backlogged TCP flows. We used a 4-hop parking lot topology, with TCP flows sourced at each successive nodes in a chain, and terminating at the final node. We used uniform link rates for a given experiment, varying the link rates from 6.5 Mb/s, 144.4 Mb/s, and 300 Mb/s in different experiments.

Our first set of experiments were performed with transmit A-MPDU aggregation disabled. Our throughput and RTT results are summarized in Figs. 6a and 6b.

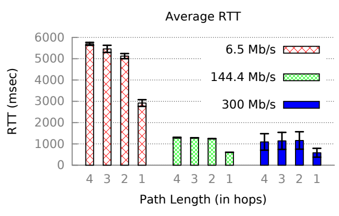
Fig. 6a shows the flow throughput dropping with increasing hop-count. Across all scenarios, the 1-hop flow obtained the highest throughput, followed by the 2-hop flow, etc. This is the well-known fairness problem in WMN [12]. We use Jain's Fairness Index (JFI) to quantify the degree of fairness in rate allocation. The JFI computed for the parking lot topology using 6.5 Mb/s, 144.4 Mb/s, and 300 Mb/s links is 0.86, 0.81, and 0.46 respectively. We observe that the rate allocation becomes more unfair with faster link rates. At these high rates, the 1-hop node can quickly build up a large TCP congestion window, saturating its buffers and starving out flows that traverse more hops. We note that some unfairness in flow rates is expected, since TCP allocates rates in proportion to the RTT. The propagation delay for a 4-flow hop is at least 4 times larger than that of a 1-hop flow, but we observe that its throughput is significantly smaller, *e.g.,* the 1-hop flow throughput is approximately 2.3×, 4.7×, and 33× the 4-hop flow throughput with the 6.5 Mb/s, 144.4 Mb/s, and 300 Mb/s link rates, respectively. This throughput imbalance persists because of the disproportionate queueing delays experienced by different flows (We note that nodes in our network do not suffer from the hidden terminal or related wireless challenges).

Next, we repeated these experiments with transmit A-MPDU aggregation enabled. Our results are shown in Fig. 6d and 6d. Experiments with 6.5 Mb/s link rates are omitted since the ath9k driver does not support transmit A-MPDU aggregation at that rate.

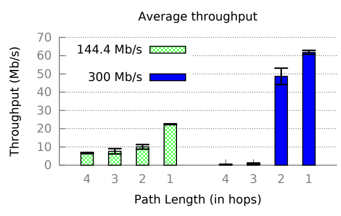
We observe that the unfairness in rate allocation persists, as expected. The JFI for 144.4 Mb/s and 300 Mb/s link rate is 0.77 and 0.50, respectively. The 1-hop flow throughput is 3.38× and 150× the 4-hop flow throughput for the 144.4 Mb/s and 300 Mb/s, respectively. This reinforces the observation that unfairness in rate allocation increases with faster link rates, as both 1-hop and 2-hop flows can quickly saturate the local node buffers. Indeed, in some of our experimental runs, the distant hop flows took a long time just to establish a TCP connections, with initial TCP setup control messages encountering full buffers along the routing path.



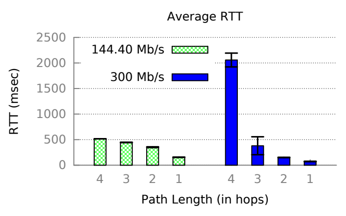
(a) Throughput without aggregation



(b) RTT without aggregation



(c) Throughput with aggregation



(d) RTT with aggregation

Fig. 6: Flow throughput and RTT for parking lot topologies

This unfairness problem may partly be addressed using buffer sizing techniques, similar to the AP buffer sizing techniques described in [6]. We are currently exploring this avenue for addressing the fairness problem in WMNs.

VI. Conclusions and Open Issues

In this paper we studied the impact of bufferbloat on wireless network performance. Using experiments on a Linux testbed with IEEE 802.11n radios, we showed that the Linux configuration with default buffer sizes can produce large queueing delays, with RTT values averaging 1700 ms for a single backlogged TCP stream on a 1-hop network with a 6.5 Mb/s wireless link rate. Multi-hop wireless networks have additional buffers at each hop. These large buffers further increase the queueing delays, with RTT values approaching 4600 ms for a 4-hop chain topology with uniform 6.5 Mb/s wireless link rates. Our queue utilization analysis shows that while RTT measurements increase with the hop count, they do not always exhibit a proportionate increase, especially at low wireless link rates where TCP's feedback mechanism is unable to saturate all buffers. We show that A-MPDU aggregation can be used to reduce RTT values, while simultaneously improving throughput. However, the RTT values still approach 350 ms over a 4-hop network. Such delays are catastrophic when queues are shared with real-time flows such as VoIP with strict latency and jitter requirements. Our analysis showed a smaller A-MPDU size in multi-hop networks; here, packets are dispersed over multiple nodes, and thus a given node may not always have the maximum number of MPDUs ready to transmit together. Clearly, there are performance tradeoffs involved here between buffer sizes and using maximum-sized A-MPDU frames that we plan to explore in future work.

Large buffer sizes can also impact the fair rate allocation in parking lot based wireless topologies. Our experiments showed that 1-hop and 2-hop flows can quickly saturate their local buffers, especially at high wireless link rates, starving distant flows. In our ongoing work, we are exploring the use of buffer sizing techniques to limit the unbridled growth of the TCP congestion window for small hop-count flows to improve flow rate fairness.

The bufferbloat community is exploring the use of Active Queue Mangagement (AQM) techniques to manage persistent queues. While AQM techniques are known to have configuration challenges, a new 'no-knobs' technique called CoDel [15] is now actively being researched. However, the current focus is on performance evaluation for cable modems and DSL networks. In future work, we plan to conduct a detailed study of CoDel in multi-hop wireless networks.

References

[1] ath9k FOSS drivers. http://wireless.kernel.org/en/users/Drivers/ath9k.
[2] Bufferbloat. http://www.bufferbloat.net.
[3] Shuttle Inc. http://www.shuttle.com.
[4] The Web10g Project. http://www.web10g.org/.
[5] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. In *Proc. of the ACM SIGCOMM '04*, pages 281–292, Sept. 2004.
[6] R. Bruno, M. Conti, and E. Gregori. Analytical modeling of TCP clients in Wi-Fi hot spot networks. In *Proc. of the IFIP Networking '04*, pages 626–637, May 2004.
[7] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden. Routers with very small buffers. In *Proc. of the IEEE INFOCOM '06*, Apr. 2006.
[8] J. Friedrich, S. Frohn, S. Gubner, and C. Lindemann. Understanding IEEE 802.11n Multi-hop Communication in Wireless Networks. In *Workshop on Wireless Network Measurements*, pages 321–326, May 2011.
[9] IEEE LAN/MAN Standards Committee. *IEEE 802.11 Wireless LAN medium access control (MAC) and physical layer (PHY) specifications*. IEEE, 2012.
[10] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance. RFC 1323, Internet Engineering Task Force, May 1992.
[11] K. Jamshaid, B. Shihada, L. Xia, and P. Levis. Buffer sizing in 802.11 wireless mesh networks. In *Proc. of the IEEE MASS '11*, pages 272–281, Oct. 2011.
[12] K. Jamshaid, P. Ward, and M. Karsten. Mechanisms for centralized flow rate control in 802.11-based wireless mesh networks. *Computer Networks*, 56(2):884–901, February 2012.
[13] H. Jiang, Y. Wang, K. Lee, and I. Rhee. Tackling bufferbloat in 3G/4G mobile networks. In *Proc. of the ACM IMC'12*, pages 329–342, Oct. 2012.
[14] T. Li, D. J. Leith, and D. Lamone. Buffer sizing in 802.11 wireless networks. *IEEE/ACM Transactions on Networking*, 19(1):156–169, Feb. 2011.
[15] K. Nichols and V. Jacobson. Controlling queue delay. *Communications of the ACM*, 55(7):42–50, July 2012.
[16] D. Skordoulis, Q. NI, H. Chen, A. Stephens, C. Liu, and A. Jamalipur. IEEE 802.11N MAC frame aggregation mechanisms for next-generation high-throughput WLANs. *IEEE Wireless Communications*, pages 40–47, February 2008.
[17] C. Villamizar and C. Song. High performance TCP in ANSNET. *SIGCOMM Computer Communications Review*, 24(5):45–60, 1994.
[18] K. Xu, M. Gerla, L. Qi, and Y. Shu. Enhancing TCP fairness in ad hoc wireless networks using neighborhood RED. In *Proc. of the ACM MobiCom '03*, pages 16–28, Sept. 2003.