Improving TCP Goodput in 802.11 Access Networks

Long Le, Sahin Albayrak, Muslim Elkotob, Ahmet Cihat Toker
Department of Electrical Engineering and Computer Science
Technische Universität Berlin, Germany

Abstract— **The widespread deployment of the IEEE 802.11 protocol has made it the de facto standard for wireless network access and allows Internet users to move freely (at least within a hotspot coverage). The convenience brought about by the IEEE 802.11 protocol is also accompanied by new technical challenges such as poor performance of the widely used transport protocol TCP due to the limited and varying bandwidth resources of the wireless medium. In this work, we seek to enhance efficiency of bandwidth usage for TCP over wireless links and improve TCP goodput. Our insight is that TCP acknowledgments can be spared on wireless links to save precious bandwidth resource of these links. We propose a simple technique that leverages the layer coordination between TCP and MAC to suppress TCP acknowledgments. Our technique is *transparent* to both TCP and MAC and *does not* require their modification or replacement.**

I. Introduction and Motivation

As wireless technologies continue to mature, they have become an integral part of the Internet. Wireless technologies such as the IEEE 802.11 protocol provide users with easy access to the Internet and have now been widely deployed at many places such as university campuses, corporate networks, hotels, coffee shops, and airports. The deployment of wireless Internet allow users to move freely (at least within a wireless hotspot coverage). Beyond serving as a last-hop link, wireless technologies such as the IEEE 802.11 protocol can also be used to set up wireless infrastructure in metropolitan areas [2], [8], [1], [9] as well as in rural areas [29].

The new opportunities opened up by wireless technologies are also accompanied with new technical challenges. First, the wireless medium has limited and varying bandwidth resources. Second, due to intermediate bit error rates and interferences between nodes [4], performance of network protocols such as TCP that were originally designed for wired networks degrade significantly on wireless networks [15]. Third, since the current IEEE 802.11 technology does not provide any mechanism to support fairness and/or quality of service, severe unfairness between flows can arise in wireless networks [27], [10], [18]. For example, when multiple wireless stations are serviced by an access point, unfairness between upstream and downstream flows occurs. This situation arises because the access point has the same probability for acquiring the wireless medium as each wireless station but must simultaneously serve multiple downstream flows.

Significant research and standardization effort has been undertaken to address the challenges mentioned above. For example, the IEEE 802.11e protocol [32] supports quality of service and allows a certain node to obtain a larger share of available bandwidth than other wireless stations. Besides IEEE 802.11e, a number of MAC protocols have been proposed to improve efficiency of bandwidth usage and address unfairness issues [3], [10], [19]. Another line of research applies the concept of the Weighted Fair Queuing (WFQ) algorithm [13] and its variants such as Self-Clocked Fair Queueing [16] or Start-Time Fair Queueing [17] to the context of wireless networks. These approaches either have an ideal centralized node that has perfect knowledge of all nodes [21], [22], [24], [26] or employ a distributed algorithm that approximates the centralized model [30], [23]. Other approaches propose modifications to TCP [20], [11] or support mechanisms for TCP [6], [27]. We note that most of these approaches require substantial modifications of either TCP or the IEEE 802.11 protocol and face significant deployment hurdles.

In this work, we investigate a new approach for improving efficiency of bandwidth usage for the widely used transport protocol TCP without any modification of either TCP or the IEEE 802.11 protocol. In fact, a salient feature of our approach is that it is *transparent* to both TCP and the 802.11 protocol and *does not* require their modification or replacement (the rationale behind our approach is that any solutions that require modifications of either operating systems or network infrastructure usually fail to obtain widespread deployment). A key observation in our approach is that TCP acknowledgment packets can be spared to save precious bandwidth resource on wireless first-hop/last-hop links.

We make the following important assumptions in this paper: (1) Wireless stations access the Internet via a WLAN access point. Thus, TCP flows (either upstream or downstream) have a *single* wireless first-hop/last-hop link between the access point and the wireless stations. Improving efficiency of bandwidth usage for TCP flows between two arbitrary nodes in ad hoc wireless networks is not addressed in this paper and is a subject of our future research. (2) It is in the interest of the access point to improve the performance of the wireless stations' TCP flows and the wireless stations can trust the access point to a certain degree. This assumption is consonant with current deployment scenarios, e.g., the owner of a wireless hotspot is interested in improving efficiency of bandwidth usage to be able to serve more customers. Further, like in the current deployment scenario, if the access point were to act selfishly or maliciously, it could simply deny providing services to the wireless stations, silently drop their packets, or sniff them. Our two assumptions allow the access point to function as a proxy for the wireless stations.

The rest of our paper is organized as follows. Section II reviews related work. Section III quantifies the overhead of TCP acknowledgment packets over the IEEE 802.11 protocol and gives the details of our approach. Section IV presents results of our simulation. Section V concludes our paper.

II. Related Work

As noted in Section I, a large amount of work has been done by the research community to address the challenges of wireless networks. Existing work falls roughly into the following categories: (1) improving MAC performance, fairness, and enhancing service differentiation, (2) designing support mechanisms for TCP, and (3) modification of TCP.

Aad and Castelluccia investigated MAC mechanisms to support service differentiation in IEEE 802.11 networks by evaluating three different schemes: variation of the contention window (each priority level uses a different backoff increment function), variation of the inter frame spacing (each priority level has a different DCF Inter-frame SpaceDIFS value), and variation of the maximum frame length [3]. Their simulations showed that variation of DIFS values obtains the best general properties and recommended it for service differentiation.

Barry et al. proposed a modified MAC that uses different minimum and maximum for backoff intervals for different traffic classes (e.g., interactive applications such as voice has higher priority than background TCP) [7]. Further, they proposed a virtual MAC (VMAC) algorithm that emulates the behavior of real MAC but makes no actual data transmission. VMAC is used to estimate the load of the wireless medium and provides hints for a distributed admision control algorithm.

The IEEE 802.11e protocol has been finalized as an extension of the IEEE 802.11 protocol to provide quality of service over wireless LANs [32]. The IEEE 802.11e protocol generalizes the IEEE 802.11 protocol and supports different traffic categories such as voice, audio, video, and data. When a wireless station has backlogged data from multiple traffic categories, it will compete for the wireless medium with the highest backlogged traffic category. Differentiation between traffic categories is realized by assigning different minimum and maximum values from which backoff intervals are chosen such that the higher the traffic category, the shorter the backoff interval is.

A large body of work investigated fair scheduling algorithms in wireless networks [21], [22], [24], [26], [30], [23]. These algorithms were derived from the Weighted Fair Queuing (WFQ) algorithm [13] and its variants such as Self-Clocked Fair Queueing [16] or Start-Time Fair Queueing [17] for the context of wireless networks. The key insight of these algorithms is that a backlogged flow unable to transmit its packets during its scheduled slot due to channel errors can be later compensated. Different algorithms differ in how compensation is done and how much compensation is allowed to enforce both short-term and long-term fairness. Further, wireless fair scheduling algorithms are either centralized [21], [22], [24], [26] or distributed [30], [23].

Casetti et al. presented TCP Westwood where the sender estimates the end-to-end available bandwidth and adjusts its transmission rate based on the arrivals of TCP ACKs [11]. The key idea is that arrivals of TCP ACKs provide better congestion indication than simply counting duplicate ACKs (such as done by TCP Reno) in the presence of lossy links.

Bakre and Badrinath proposed Indirect TCP (I-TCP) to support host mobility [5]. I-TCP divides a TCP connection into two separate connections at the base station: a regular TCP connection between the fixed host and the base station and a specialized TCP connection between the base station and the mobile host. When a handoff occurs, retransmission timers of the specialized TCP connection are cleared and the connection enters slow start to obtain new information about the network conditions.

Balakrishnan et al. introduced a *snoop agent* at the base station that monitors TCP packets in both directions and caches TCP data packets that were sent across the wireless link but have not yet been acknowledged [6]. The snoop agent can detect a packet loss by the expiration of a timer or by the arrival of a number of duplicate ACKs. The snoop agent retransmits the lost packet and suppresses duplicate ACKs.

Chakravorty et al. improved HTTP performance over GPRS links by using a transparent TCP proxy [12]. The TCP proxy divides a TCP connection between a wireless station and a fixed host into two TCP connections: one between the wireless node and the proxy and the other between the proxy and the fixed host. The TCP connection between the proxy and the wireless node is optimized for GPRS links to avoid slow start and further growth of the congestion window beyond the bandwidth delay product. Further, since packet losses on GPRS links are usually due to radio losses or cell reselections, congestion control mechanisms such as halving the congestion window are unnecessary and can be avoided.

Pilosof et al. investigated fairness issues between upstream and downsream TCP flows at a base station through analysis and simulation [27]. They observed that the unfairness in throughput ratio between upstream and downstream TCP flows can be as high as 800. This unfairness arises from the fact that the base station has to forward packets for multiple downstream TCP flows but only has an equal chance to acquire the wireless medium as the wireless stations. Pilosof et al. proposed to improve fairness between upstream and downstream TCP flows by manipulating the receiver's advertised window in TCP ACKs (and thus throttle upstream TCP flows that are using more bandwidth than their fair share).

With the exceptions of the work by Balakrishnan et al., Chakravorty et al., and Pilosof et al., other work requires modification of either TCP or the network infrastructure and are thus unlikely to be deployed in near future. The motivation for our work is that we need a *transparent* solution that can interact with TCP and IEEE 802.11 networks.

III. Improving TCP Goodput in 802.11 Access Networks

Our approach is based on the observation that TCP acknowledgment packets (ACK packets) can be spared to save precious bandwidth resource on the wireless link. Although these TCP ACK packets are small, they incur a large overhead at the

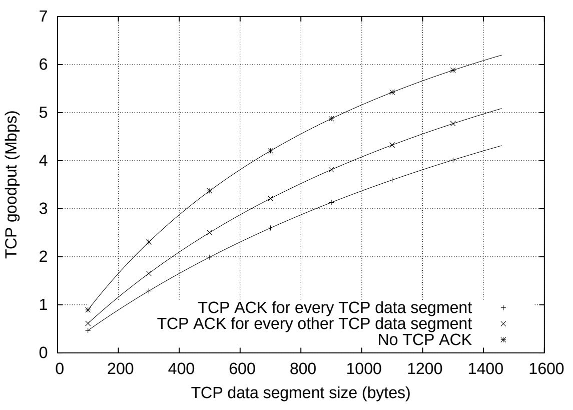

Fig. 1. TCP goodput as a function of TCP segment size in an 802.11b access network

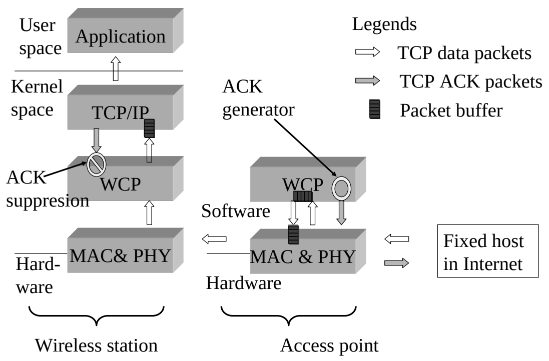

Fig. 2. Architectural Overview

link layer because the RTS/CTS handshake at the link layer has to be completed for the transmission of *each* IP packet. Further, the IEEE 802.11 standard dictates that the RTS and CTS frames be transmitted at the base rate [31]. Thus, the higher the rate the wireless link is operated at, the larger the overhead of the RTS/CTS handshake for a transmission of an IP packet is on the wireless link. The rest of this Section is organized as follows. The overhead for TCP ACK packets is quantified in Section III-A. Section III-B presents the overall system architecture of our approach. Sections III-C and III-D discuss the details of our approach for downstream and upstream TCP flows.

A. Quantifying the Overhead of TCP ACK packets

Let $bRate$ and $cRate$ be the basic and channel data rate, rts, cts, and M_{ack} be the sizes of the RTS, CTS, and ACK frame, $sifs$ and $difs$ be the SIFS and DIFS interval in the IEEE 802.11 protocol. Further, let H_{tcp} and H_{mac} be the header sizes of a TCP segment and a MAC frame (including preamble). The transmission time for a TCP data segment of size s is computed as

$$T_{data} = difs + 3 \times sifs + \frac{rts + cts}{bRate} + \frac{s + H_{tcp} + H_{mac} + M_{ack}}{cRate} \tag{1}$$

The transmission for a TCP ACK packet is computed as

$$T_{ack} = difs + 3 \times sifs + \frac{rts + cts}{bRate} + \frac{H_{tcp} + H_{mac} + M_{ack}}{cRate} \tag{2}$$

Let n be the average number of TCP ACKs generated by a TCP receiver of a TCP data segment. If all TCP ACKs are suppressed, $n = 0$. If a TCP receiver sends a TCP ACK for *every other* TCP data segment, $n = 0.5$. If a TCP receiver sends a TCP ACK for *every* TCP data segment, $n = 1$. The goodput of a long-lived TCP flow is computed as

$$Goodput_{TCP} = \frac{s}{T_{data} + n \times T_{ack}} \tag{3}$$

Figure 1 depicts the goodput of a long-lived TCP flow over an 802.11b network as a function of the TCP data segment size s (we use a typical setting of an 802.11b network: $bRate = 1\ Mbps$, $cRate = 11\ Mbps$, $sifs = 10\ \mu s$, $difs = 50\ \mu s$, $rts = 44\ bytes$, $cts = 40\ bytes$, $H_{mac} = 40\ bytes$, and $H_{tcp} = 40\ bytes$). As expected, TCP goodput increases with the TCP data segment size since the overhead (RTS/CTS and protocol overhead) is ameliorated. Further, as can be seen in Figure 1, TCP goodput can be improved by approximately 50% if TCP ACKs are suppressed (compared to TCP receiver's policy of acknowledging every TCP data segment). When compared to TCP receiver's policy of acknowledging every other TCP data segment, TCP goodput with ACK suppression can be improved by approximately 20%.

B. Architectural Overview

We present a simple technique to remove the overhead of TCP ACK packets by leveraging the layer coordination between TCP and MAC in 802.11 wireless networks. Our technique exploits the fact that the IEEE 802.11 protocol already implements a semi-reliable transmission: each data frame encapsulating an IP packet transmitted on the wireless medium is succeeded by an ACK frame that acknowledges the successful transmission of the data frame. Thus, TCP ACK packets can be suppressed and acknowledgments for TCP data packets that were successfully transmitted across the wireless link can be implicitly provided by using the 802.11 ACK frames. The absence of an ACK frame indicates that the transmission of the data frame was unsuccessful and a retransmission is necessary. We note that transmissions provided by the IEEE 802.11 protocol are semi-reliable because retransmissions for RTS and a short data frame are performed up to $ShortRetryLimit$ times and for a long data frame up to $LongRetryLimit$ times. After that, the data frame is dropped and higher layer is notified of the failed transmission. $ShortRetryLimit$ and $LongRetryLimit$ are configurable parameters with default values of 7 and 4 respectively [31].

To remove the overhead caused by TCP ACK packets on wireless links, we rely on the WLAN access point to act as a proxy for the wireless stations. The access point buffers

TCP data packets for wireless stations. Further, for upstream TCP flows, i.e., when wireless stations are TCP senders, the access point suppresses TCP ACK packets from fixed hosts in Internet. For downstream TCP flows, i.e., when wireless stations are TCP receivers, the access point generates TCP ACK packets for the wireless stations and passes on these packets to fixed hosts in the Internet. (We note that our technique does not prevent or adversely affect TCP performance in scenarios where the other communicating host is also a wireless station. However, for ease of presentation, we assume that the other communicating host connects to fixed networks in the Internet.)

One can argue that our technique weakens the end-to-end semantics because TCP data packets may be acknowledged prematurely before they actually reach the final destinations. However, we note that the end-to-end semantics can be retained to a certain degree by refraining from acknowledging TCP FINs prematurely (i.e., TCP FINs have to be exchanged end-to-end).

We introduce a new component called Wireless Coordination Protocol (WCP) that sits between the IP and the MAC layer at the access point and the wireless stations. Fig. 2 illustrates the overall system architecture of our approach. Our techniques assume that WCP can intercept and suppress outgoing TCP ACK packets from the IP layer when necessary. In BSD, Linux and Windows systems, this can actually be achieved quite easily by implementing an intermediate layer between the network device drivers and IP. Further, we assume a close coordination between the WCP and the MAC layer. In particular, we assume that WCP will be informed by the MAC layer about the successful or failed transmission of a TCP data packet. This assumption is necessary for the access point to generate TCP ACK packets for fixed hosts in the Internet (for downstream TCP flows). Further, this assumption is also necessary for the access point and the wireless stations to initiate a retransmission of a TCP data packet that was dropped by the link layer after a number of failed attempts. We note that this assumption *does not* require any modification to the MAC protocol - at most, it only requires a modification to the MAC driver. This assumption is actually quite realistic since many modern wireless NICs and their associated drivers offer fine-grained control over the hardware [25].

C. ACK Suppressions in a Downstream TCP Flow

Details of our approach for a downstream TCP flow are depicted in Fig. 3. The access point caches TCP data packets originating from fixed hosts in the Internet in a packet buffer and subsequently transmits these packets to wireless stations. When a TCP data packet is successfully transmitted across the wireless link, the access point removes the TCP data packet from its internal buffer, generates a TCP ACK packet for the TCP data packet, and sends the TCP ACK packet to the fixed host. Outgoing TCP ACK packets at the wireless stations are intercepted and suppressed by the WCP layer.

When notified of the failed transmission of a TCP data packet, WCP at the access point passes down that TCP data packet to the MAC layer again and initiates its retransmission.

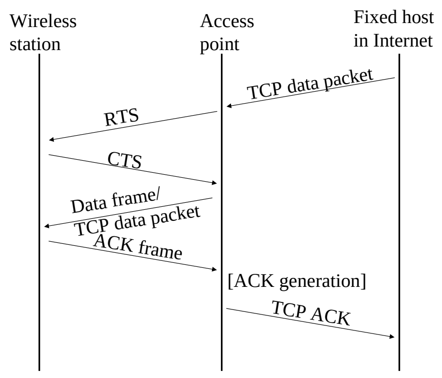

Fig. 3. ACK suppression in a downstream TCP flow

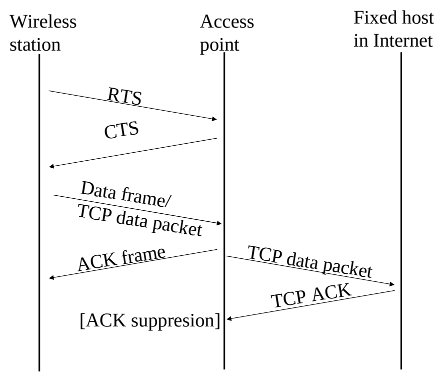

Fig. 4. ACK suppression in an upstream TCP flow

Since our overall goal is to improve TCP performance without requiring modifications of TCP implementation or the IEEE 802.11 protocol, we need to be careful in this case. On the one hand, WCP should be able to pass down several outstanding TCP data packets from a TCP flow to the MAC layer. The rationale behind this is that we want to have several packets buffered at the MAC layer so that processing time at the WCP layer would not result in missed opportunities for transmission at the MAC layer and would not degrade the link throughput. On the other hand, since multiple TCP data packets are buffered and transmitted at the MAC layer in a (typically) FIFO manner, the retransmission of a TCP data packet would cause the reordering of TCP data packets at the wireless stations and trigger duplicate TCP ACKs. For this reason, WCP at the wireless stations must intercept and suppress the duplicate TCP ACKs to avoid using precious bandwidth of the wireless link.

Since TCP data packets originating from fixed hosts may be lost or reordered before arriving at the wireless link, the access point must send duplicate TCP ACK packets on behalf of the wireless stations when it detects out-of-order TCP data packets. These TCP ACK packets are necessary for the fixed hosts to retransmit their lost TCP data packets. However, the access point does not have to buffer out-of-order TCP data packets and send them to the wireless stations in-order. Rather, the access point simply forwards TCP data packets to the wireless stations in the order that they arrive (reordering can be done by local TCP receivers and suppression of duplicate TCP ACKs can be done by WCP at the wireless stations).

D. ACK Suppressions in an Upstream TCP Flow

Details of our approach for an upstream TCP flow are illustrated in Fig. 4. The access point caches TCP data packets originating from wireless stations in a packet buffer and forwards them to fixed hosts in the Internet. WCP at the access point performs the following tasks for upstream TCP flows. (1) It intercepts and suppresses TCP ACK packets from fixed hosts in the Internet to save bandwidth resource of the wireless link. Further, it removes cached TCP data packets that are acknowledged by TCP ACKs from fixed hosts from its packet buffer. (2) Since TCP data packets may be lost in the Internet, WCP at the access point has to detect (and drop) duplicate TCP ACKs from fixed hosts in the Internet, and retransmits the lost TCP data packets (as indicated by the duplicate TCP ACKs). Further, since duplicate TCP ACKs from fixed hosts may never arrive at the access point, WCP needs to maintain retransmission timers for the transmitted TCP data packets (in the same manner as a TCP sender would do it). When a timer expires, the TCP data packet with that timer is retransmitted. (3) WCP at the access point needs to maintain a congestion window that dictates the maximum number of TCP data packets allowed for an upstream TCP flow to avoid causing congestion in the Internet. This congestion window is updated by using the TCP ACK packets from fixed hosts in the Internet in the same manner as TCP would update a congestion window. In other words, since the access point acts as a proxy for the wireless stations, it implements most features of a regular TCP stack.

The WCP layer at the wireless stations generates a TCP ACK packet for each TCP data packet that is successfully transmitted across the wireless link and requests the IP layer to pass the TCP ACK packet up to the TCP layer.

A problem that could ensue at the wireless stations is that since all TCP data packets are acknowledged by the WCP layer, TCP could increase its congestion window to a large value and would have too many outstanding TCP data packets. Consequences of this potential problem are two-fold. (1) It causes unnecessary buffering at the wireless stations that not only consumes memory resources but also increases end-to-end latency (and thus has an adverse impact on interactive applications such as web browsing). (2) It can cause the internal buffer at the IP layer to overflow (the IP layer's packet buffer is typically set to about 50 packets on Linux and BSD systems that we know of) that results in loss of TCP data packets and has an adverse effect on TCP performance.

There are several methods to prevent the problem mentioned above. (1) The simplest method is to monitor the number of outstanding TCP data packets of an upstream TCP flow at the WCP layer at the wireless stations. If this number exceeds a certain threshold, say 20 packets, WCP acts like a RED router [14] and drops outgoing TCP data packets probabilistically. We believe that this method is too coarse. (2) WCP can generate three duplicate TCP ACKs and request the IP layer to pass these packets to TCP. This would cause the TCP sender at a wireless station to reduce its transmission rate. Since this method would cause retransmission of TCP data packets, WCP must intercept outgoing TCP data packets, detect and drop the duplicate TCP data packets. This method appears to be rather complicated. (3) If ECN [28] is supported by the TCP sender and receiver, a more elegant solution for WCP is to set the Congestion Window Reduced (CWR) bit in the TCP header of a TCP ACK packet that it generates when the number of outstanding TCP data packets exceeds a certain threshold. In this method, WCP acts like an ECN-enabled RED router. However, this method requires ECN support at both end systems. (4) WCP generates a TCP ACK packet where the receiver window is set to 0 to throttle the local TCP sender. Later, when the buffer of outgoing TCP data packets shrinks, WCP can generate another TCP ACK packet with a non-zero receiver window to allow the local TCP sender to transmit its packets again. This is the method that we use in our approach.

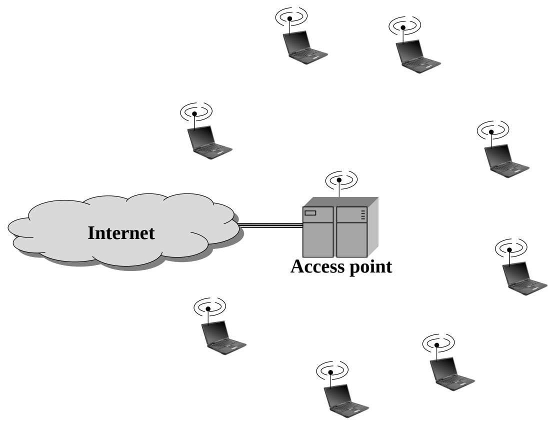

Fig. 5. Simulation topology.

IV. Simulation Results

We performed simulations in *ns-2* to quantify the effects of ACK suppression. Our simulation topology is shown in Fig. 5 and models a scenario where N wireless stations are placed equi-distant from the access point. The wireless network is 802.11b with a base rate of 1 Mbps and a channel data rate of 11 Mbps. The TCP segment size is set to 1460 bytes. The access point is connected to the Internet via a 100-Mbps uplink with a one-way propagation delay of 20 milliseconds. All experiments ran for 30 minutes but results of the first 10 minutes were discarded to eliminate the start-up phase. We vary the number of upstream and downstream TCP flows and show our results in Figres 6 and 7. Our results demonstrate the performance improvement of TCP ACK suppression is approximately 50% in comparison with the receiver's policy of acknowledging every TCP data segment and approximately 20% in comparison with the receiver's policy of acknowledging every other TCP data segment.

V. Summary and Conclusions

In this paper, we argued that TCP ACK packets on wireless links can be spared to save precious bandwidth on these links.

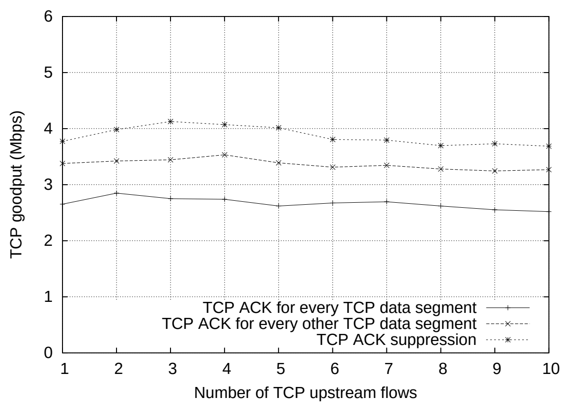

Fig. 6. Total TCP goodput as a function of number of upstream TCP flows

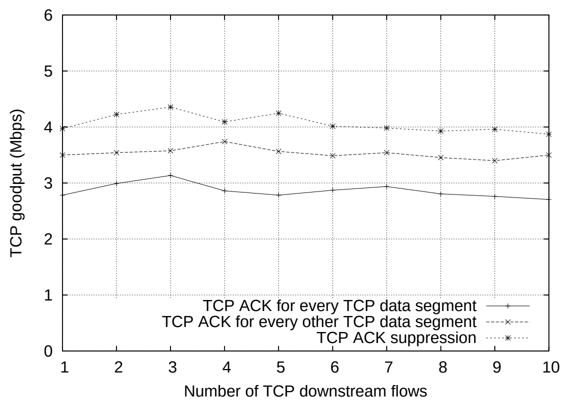

Fig. 7. Total TCP goodput as a function of number of downstream TCP flows

We quantified the overhead of TCP ACK packets on these links to be as high as 50%. We presented a simple technique that leverages layer coordination between TCP and MAC to spare TCP ACK packets and improve efficient bandwidth usage of wireless links. Our technique has access points act as a proxy for TCP senders or receivers at wireless stations. Unlike most existing work, that improves TCP raw throughput, our work seeks to improve TCP goodput by suppressing TCP ACKs over wireless links. Further, an attractive feature of our technique is that it is transparent to both TCP and MAC and does not require their modification.

References

[1] NYCWireless, 2006. `http://www.nycwireless.net/`.

[2] Seattle wireless, 2006. `http://www.seattlewireless.net/`.

[3] Imad Aad and Claude Castelluccia. Differentiation mechanisms for IEEE 802.11. In *IEEE INFOCOM*, April 2001.

[4] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris. Link-level measurements from an 802.11b mesh network. In *ACM SIGCOMM*, August 2004.

[5] Ajay Bakre and B. R. Badrinath. I-TCP: Indirect TCP for mobile hosts. In *IEEE International Conference on Distributed Computing Systems*, May 1995.

[6] Hari Balakrishnan, Srinivasan Seshan, Elan Amir, and Randy H. Katz. Improving TCP/IP performance over wireless networks. In *ACM MobiCom*, November 1995.

[7] Michael Barry, Andrew Campbell, and Andras Veres. Distributed control algorithms for service differentiation in wireless packet networks. In *IEEE INFOCOM*, April 2001.

[8] BARWN. Bay area research wireless network, 2006. `http://www.barwn.org/`.

[9] John Bicket, Daniel Aguayo, Sanjit Biswas, and Robert Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *ACM MobiCom*, August 2005.

[10] M. Bottigliengo, C. Casetti, C.-F. Chiasserini, and M. Meo. Short-term fairness for TCP flows in 802.11b WLANs. In *IEEE INFOCOM*, March 2004.

[11] Claudio Casetti, Mario Gerla, Saverio Mascolo, M. Y. Sanadidi, and Ren Wang. TCP Westwood: end-to-end congestion control for wired/wireless networks. *Wireless Networks*, 8, September 2002.

[12] Rajiv Chakravorty, Sachin Katti, Jon Crowcroft, and Ian Pratt. Flow aggregation for enhanced TCP over wide area wireless. In *IEEE INFOCOM*, April 2003.

[13] Alan Demers, Srinivasan Keshav, and Scott Shenker. Analysis and simulation of a fair queueing algorithm. In *ACM SIGCOMM*, September 1989.

[14] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[15] Zhenghua Fu, Petros Zerfos, Haiyun Luo, Songwu Lu, Lixia Zhang, and Mario Gerla. The impact of multihop wireless channel on TCP throughput and loss. In *IEEE INFOCOM*, April 2003.

[16] S. Jamaloddin Golestani. A self-clocked fair queueing scheme for broadband applications. In *IEEE INFOCOM*, June 1994.

[17] Pawan Goyal, Harrick Vin, and Haichen Cheng. Start-time fair queueing: a scheduling algorithm for integrated services packet switching networks. In *ACM SIGCOMM*, August 1996.

[18] Martin Heusse, Franck Rousseau, Gilles Berger-Sabbatel, and Andrzej Duda. Performance anomaly of 802.11b. In *IEEE INFOCOM*, April 2003.

[19] Martin Heusse, Franck Rousseau, Romaric Guillier, and Andrzej Duda. Idle sense: An optimal access method for high throughput and fairness in rate diverse wireless LANs. In *ACM SIGCOMM*, August 2005.

[20] Hung-Yun Hsieh, Kyu-Han Kim, Yujie Zhu, and Raghupathy Sivakumar. A receiver-centric transport protocol for mobile hosts with heterogeneous wireless interfaces. In *ACM MobiCom*, September 2003.

[21] Songwu Lu, Vaduvur Bharghavan, and Rayadurgam Srikant. Fair scheduling in wireless packet networks. In *ACM SIGCOMM*, September 1997.

[22] Songwu Lu, Thyagarajan Nandagopal, and Vaduvur Bharghavan. A wireless fair service algorithm for packet cellular networks. In *ACM MobiCom*, August 1998.

[23] Haiyun Luo, Songwu Lu, and Vaduvur Bharghavan. A new model for packet scheduling in multihop wireless networks. In *ACM MobiCom*, September 2000.

[24] Thyagarajan Nandagopal, Songwu Lu, and Vaduvur Bharghavan. A unified architecture for the design and evaluation of wireless fair queueing algorithms. In *ACM MobiCom*, August 1999.

[25] Michael Neufeld, Jeff Fifield, Christian Doerr, Anmol Sheth, and Dirk Grunwald. SoftMAC—flexible wireless research platform. In *ACM Workshop on Hot Topics in Networks*, November 2005.

[26] T. S. Eugene Ng, Ion Stoica, and Hui Zhang. Packet fair queueing algorithms for wireless networks with location-dependent errors. In *IEEE INFOCOM*, March 1998.

[27] Saar Pilosof, Ramachandran Ramjee, Danny Raz, Yuval Shavitt, and Prasun Sinha. Understanding TCP fairness over wireless LAN. In *IEEE INFOCOM*, April 2003.

[28] K. K. Ramakrishnan, Sally Floyd, and David L. Black. The addition of explicit congestion notification (ECN) to IP. RFC 3168, September 2001.

[29] Bhaskaran Raman and Kameswari Chebrolu. Revisiting MAC design for an 802.11-based mesh network. In *ACM Workshop on Hot Topics in Networks*, November 2004.

[30] Nitin Vaidya, Paramvir Bahl, and Seema Gupta. Distributed fair scheduling in a wireless LAN. In *ACM MobiCom*, September 2000.

[31] IEEE 802.11 WG. Wireless lan medium access control (MAC) and physical layer (PHY) specifications. IEEE 802.11, November 1999.

[32] IEEE 802.11 WG. Draft supplement to IEEE standard 802.11-1999: Medium access control (MAC) enhancements for quality of service (QoS). IEEE 802.11e D6.0, November 2003.