Gateway Adaptive Pacing for TCP across Multihop Wireless Networks and the Internet*

Sherif M. ElRakabawy, Alexander Klemm, and Christoph Lindemann
University of Leipzig
Department of Computer Science
Augustusplatz 10-11
04109 Leipzig Germany
http://rvs.informatik.uni-leipzig.de/

ABSTRACT

In this paper, we introduce an effective congestion control scheme for TCP over hybrid wireless/wired networks comprising a multihop wireless IEEE 802.11 network and the wired Internet. We propose an adaptive pacing scheme at the Internet gateway for wired-to-wireless TCP flows. Furthermore, we analyze the causes for the unfairness of oncoming TCP flows and propose a scheme to throttle aggressive wired-to-wireless TCP flows at the Internet gateway to achieve nearly optimal fairness. Thus, we denote the introduced congestion control scheme TCP with Gateway Adaptive Pacing (TCP-GAP). For wireless-to-wired flows, we propose an adaptive pacing scheme at the TCP sender. In contrast to previous work, TCP-GAP does not impose any control traffic overhead for achieving fairness among active TCP flows. Moreover, TCP-GAP can be incrementally deployed because it does not require any modifications of TCP in the wired part of the network and is fully TCP-compatible. Extensive simulations using ns-2 show that TCP-GAP is highly responsive to varying traffic conditions, provides nearly optimal fairness in all scenarios and achieves up to 42% more goodput than TCP NewReno.

Categories and Subject Descriptors

C.2.0 [**Computer Communication Networks**]: General – *data communications.*

General Terms: Algorithms, Performance, Design.

Keywords: Wireless network protocols, Ad hoc networks, Performance evaluation, TCP congestion control for hybrid wireless/wired networks.

1 INTRODUCTION

Multihop wireless networks can effectively be utilized to "opportunistically" extend the range of wireless LANs and for the rapidly emerging wireless mesh networks [3]. Common Internet applications such as Web browsing, e-mail and file transfer over such hybrid wireless/wired networks require TCP as underlying protocol for reliable data transfer. A key problem for TCP over hybrid wireless/wired networks lies in the different characteristics of multihop wireless networks and the wired Internet: in multihop wireless networks most losses experienced by TCP are due to packet drops at the IEEE 802.11 link layer of intermediate nodes. Hidden terminal and exposed terminal effects are the reason for these packet drops in multihop networks [11]. In contrast, in the Internet almost all packet losses are due to buffer overflows at routers.

One solution for this problem lies in splitting the TCP connection at the node interfacing the wired and wireless part of the network, denoted as the Internet gateway. In such a split-connection approach, a specialized transport protocol like ATP [15] may run in the wireless part whereas the wired part uses standard TCP, e.g. TCP NewReno. However, a straightforward split-connection approach does not preserve the end-to-end semantics of TCP and requires complicated handover procedures in case of mobility [4]. Another solution lies in employing TCP NewReno in hybrid wireless/wired networks and performing modifications in all mobile devices of the wireless network, either on the link layer such as link layer RED [11] or on the network layer such as neighborhood RED [17]. These approaches retain the end-to-end semantics of TCP, though such an approach cannot be incrementally deployed since it requires modifications on all wireless devices.

In this paper, we introduce an effective congestion control scheme for TCP over hybrid wireless/wired networks comprising a multihop wireless IEEE 802.11 network and the wired Internet. Important classes of such networks constitute wireless mesh networks comprising mesh clients and mesh routers connected to the Internet as well as ad hoc networked mobile devices (laptops, PDAs, etc.) as opportunistic extensions to the Internet. For the effective operation of TCP over such hybrid networks, we propose to distinguish the direction of the TCP flow: For wired-to-wireless TCP flows, we introduce an adaptive pacing scheme at the Internet gateway. For wireless-to-wired flows, building upon [9], we propose an adaptive pacing scheme at the TCP sender. Furthermore, we analyze the causes for the unfairness of oncoming TCP flows in multihop wireless networks where both wired-to-wireless as well as wireless-to-wired TCP flows pass through the Internet gateway. Such unfairness was previously observed in [16] and [18]. Subsequently, we show how to throttle aggressive wired-to-wireless TCP flows at the Internet gateway to achieve nearly optimal fairness for such scenarios. Thus, we denote the introduced congestion control scheme *TCP with Gateway Adaptive Pacing (TCP-GAP)*. In contrast to previous work [12], [17], TCP-GAP does not impose any control traffic overhead for achieving fairness among active TCP flows. Moreover, TCP-GAP can be incrementally deployed, since it does not require any modifications of TCP in the wired part of the network. TCP-GAP is also fully TCP-compatible and preserves TCP-friendliness because TCP-GAP does not allow more packets to be transmitted than the current TCP window size permits.

* This work was funded in part by the German Research Council (DFG) under Grant Li-645/12-2.

We evaluate both the steady-state as well as the transient behavior of TCP-GAP using ns-2 simulation [10] with IEEE 802.11b, where we deploy scenarios describing different node topologies. The results show that TCP-GAP significantly improves both fairness and end-to-end goodput in hybrid wireless/wired networks. In fact, TCP-GAP provides excellent fairness in almost all scenarios and achieves up to 42% more goodput than TCP NewReno. In this paper, we consider scenarios with FTP-like traffic, where TCP flows are backlogged. In a further simulation study not included due to space limitations, we also consider scenarios with HTTP-like traffic which show that TCP-GAP even achieves up to 70% more goodput than TCP NewReno.

The remainder of this paper is organized as follows. Section 2 summarizes related work on TCP for hybrid wireless/wired networks and wireless mesh networks. Section 3 specifies the considered class of wireless/wired networks, for which TCP-GAP is designed. In Section 4, we introduce the congestion control scheme of TCP-GAP and present pseudo code to outline its implementation. A comprehensive performance study of TCP-GAP versus TCP NewReno is presented in Section 5. Finally, concluding remarks are given.

2 RELATED WORK

Several TCP enhancements (e.g. [9], [11], [17]) and new transport protocols such as ATP [15] were proposed for multihop wireless networks. However, only little work focused so far on improving fairness and performance of TCP over hybrid wireless/wired networks comprising a multihop wireless IEEE 802.11 network and the Internet.

In [9], we introduced TCP with Adaptive Pacing (TCP-AP) for multihop wireless networks without connection to the wired Internet. TCP-AP implements rate-based packet transmissions within the TCP congestion window. We showed how a TCP sender could adapt its transmission rate close to the optimum using an estimate of the four-hop propagation delay and the coefficient of variation of recently measured round-trip times.

Consistent with [9], we propose an adaptive pacing scheme at the TCP sender. Contrary to [9], we consider hybrid wireless/wired networks which possess different characteristics than pure multihop wireless networks and require novel approaches for improving TCP performance. Furthermore, we propose an effective solution for the unfairness problem between oncoming TCP flows spanning the wireless and wired domains of the hybrid network.

Yang et al. [18] proposed a pacing scheme at the IP layer to improve TCP fairness in hybrid wireless/wired networks. They derived the pacing rate by the minimum transmission delay observed for this node, the most recent transmission delay and a random delay. Their scheme throttles TCP flows and prevents TCP senders from transmitting too aggressively against competing flows. However, the derivation of the pacing rate in [18] is static and cannot adapt to changing network conditions; i.e., may unnecessarily throttle TCP flows. Furthermore, this approach does not distinguish between different TCP flows passing through the same wireless node.

In contrast to [18], TCP-GAP employs adaptive pacing rather than static pacing. In fact, TCP-GAP continuously determines its pacing rate by measuring the four-hop propagation delay of TCP packets and the contention on the network path. Thus, TCP-GAP does not lead to unnecessary goodput degradation if there is no contention between active flows. Furthermore, we also evaluate a considerably larger number and more realistic network topologies than [18]. Beyond [18], we show how to achieve fairness for oncoming TCP flows over a hybrid wireless/wired network.

Gambiroza et al. [12] studied TCP performance and fairness in multihop wireless networks comprising numerous wireless relay nodes (there called Transit Access Points, TAPs) and a connection to the wired Internet. They introduced TAP-fairness to characterize the idealized goodput and fairness objective for such networks and proposed a distributed link layer algorithm for achieving TAP-fairness among active TCP flows. TAP-fairness is tailored to wireless mesh networks and differs from both max-min fairness and proportional fairness. The distributed link layer algorithm for achieving TAP-fairness requires to periodically propagate the offered load and link capacities among all TAPs resulting in a significant amount of control traffic.

TCP-GAP constitutes a modification on the transport layer rather than modification on the link layer as [12]. TCP-GAP employs an adaptive pacing scheme at wireless TCP senders and the Internet gateway using an effective estimation of the four-hop propagation delay and the contention on the network path rather than measuring offered load and estimating the link capacity at each wireless relay node as [12]. In contrast to [12], TCP-GAP does not require any control traffic for achieving fairness among active TCP flows, though; we consider max-min fairness of TCP flows rather than TAP-fairness.

Mascolo et al. [13] proposed a sender-side bandwidth estimation technique for TCP over cellular mobile networks denoted as TCP Westwood to distinguish between packet losses due to buffer overflow and due to wireless losses. Akan et al. [2] proposed an adaptive transport layer suite for the next-generation wireless Internet, which deploys an adaptive congestion control method in order to account for the characteristics of the different wireless environments. In contrast to [2] and [13], we consider hybrid wireless/wired networks, in which the wireless part comprises of a multihop IEEE 802.11 network. Moreover, TCP-GAP aims at reducing performance degradation and improving fairness due to hidden and exposed terminals rather than at helping TCP to distinguish between packet losses due to buffer overflows and wireless losses.

3 CONSIDERED CLASS OF NETWORKS

We consider IEEE 802.11 multihop wireless networks which are connected through one or several fixed gateway nodes to the wired Internet. These gateway nodes are denoted as Internet gateways. Each Internet gateway has at least two network interfaces. One of them is a wireless IEEE 802.11 interface operating in ad hoc mode. The wireless subnet can be considered as an independent network running AODV [14] or other routing protocols such as ETX [7] or ETT [8] as its own routing protocol. Figure 1 illustrates the considered class of wireless/wired networks. Note that the network architecture illustrated in Figure 1 can be considered both as an opportunistic extension to the Internet with (negligible) pedestrian mobility and as an unplanned, single-radio wireless mesh network (e.g. a community network), in which some mesh routers have Internet connection.

In order to simplify the analysis of the impact of the hidden and exposed terminal effects [11], we mostly consider regular network topologies where the distance between the wireless nodes is 200m. In fact, the hidden terminal problem can even occur to a larger extend in topologies with irregular node placement. This is due to the fact that the ideal case with inter-node distances of 200 meters (given a wireless transmission range of 250 meters) roughly minimizes the number of hops necessary for a given spatial distance to an Internet gateway. With irregular node placement, the number of hops to the Internet gateway would be potentially larger resulting in even more hidden terminals. Thus, our setup constitutes a kind of

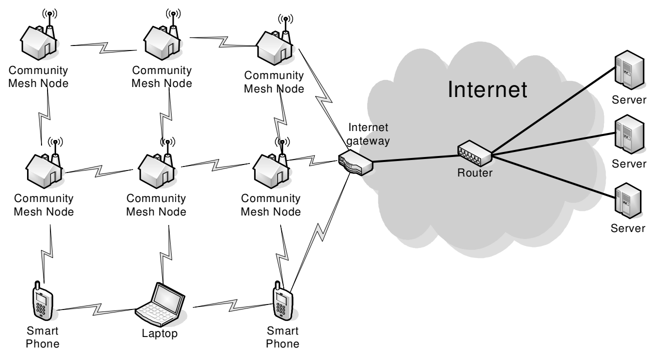


Figure 1. Targeted network architecture: Ad-hoc extension to the Internet or unplanned, single-radio wireless mesh networks

lower bound for the number of hidden terminals. Nevertheless, we also consider a random topology with irregular node placement in order to verify the applicability of our approach in such environments.

Conventional ad hoc routing protocols such as AODV [14] use the minimum hop count as routing metric. For wireless mesh networks, novel routing metrics like the expected transmission count (ETX) [7] and the expected transmission time (ETT) [8] have been proposed. These routing protocols can achieve a higher capacity in wireless mesh networks due to finding higher quality routes than routing protocols like AODV. Nevertheless, since choosing another route from source to destination cannot totally prevent hidden terminals in multihop wireless networks, these specialized routing protocols are complementary to improvements of TCP. Thus, such routing protocols tailored to wireless mesh networks may well be combined with TCP improvements such as TCP-GAP.

4 THE TCP GATEWAY ADAPTIVE PACING SCHEME

4.1 Motivation and Rationale of Gateway Adaptive Pacing

To improve TCP fairness and goodput for TCP connections across multihop IEEE 802.11 and wired networks, we propose to employ a rate-based packet scheduling within the TCP congestion window in the wireless domain while preserving the traditional TCP variant (i.e. TCP NewReno) in the wired Internet. Thus, this approach decouples the wireless part of the hybrid network from the wired part while preserving the end-to-end semantics of TCP. We achieve this transparent decoupling by adding some transport layer functionality to the IP layer at the Internet gateway.

In contrast to TCP pacing [1], the adaptive pacing approach sets the transmission rate adaptively based on the spatial reuse constraint of multihop IEEE 802.11 networks and the contention on the network path of the connection. The spatial reuse constraint [11] is accounted by considering the four-hop propagation delay (*FHD)* of TCP packets. *FHD* describes the time elapsed between transmitting a TCP packet by the TCP source node and receiving the packet at the node which lies four hops apart from the source node along the path to the destination. This measure can be estimated by measuring the round trip times (RTT) of TCP packets as well as the number of hops of the network path. The contention on the network path of the TCP connection can be estimated by measuring the variation of recent RTT samples using the coefficient of variation cov_{RTT}. In summary, the adaptive transmission rate R computed by the TCP sender is given by [9]:

$$R = \frac{1}{\widehat{FHD} \cdot (1 + 2cov_{RTT})} \qquad (1)$$

where

$$\widehat{FHD} = \alpha \cdot \widehat{FHD}_{old} + (1-\alpha) \cdot FHD \qquad (2)$$

with smoothing factor $\alpha = 0.7$.

Note that the adaptive pacing algorithm aims at improving TCP performance in the wireless domain and thus has to be implemented at the entry point of a TCP connection into the wireless network. That is, for connections spanning across wireless and wired networks we distinguish the two cases:

(1) the TCP source is a wireless device and the TCP destination resides in the wired Internet; denoted as wireless-to-wired flows.

(2) the TCP source resides in the wired Internet and the TCP destination is a wireless device; denoted as wired-to-wireless flows.

In case (1) we deploy adaptive pacing at the TCP source, whereas in case (2) we deploy adaptive pacing on IP layer at the Internet gateway. In the following subsections we will discuss the proposed adaptive pacing scheme in detail with respect to cases (1) and (2).

4.2 Gateway Adaptive Pacing for Wireless-to-Wired Flows

We consider the case where a wireless node constitutes the TCP source and a host in the wired domain constitutes the TCP destination. Figure 2 illustrates such scenario where an FTP flow runs from the wireless node A over multiple intermediate hops through the Internet gateway IG to the wired host B. Throughout this paper, wireless relay nodes are denoted by RL<i> whereas wired routers are denoted by RT<i>.

To improve TCP performance in the wireless part, we employ adaptive pacing at the wireless TCP source A. Note that conventionally measured RTT values describe the complete round-trip time of the packets crossing both the wireless and the wired parts of the network. However, for deriving proper estimates for *FHD* and cov_{RTT}, we only need the packet RTT in the wireless part, i.e. the time taken for a TCP packet to be forwarded from A to IG plus the time taken for the corresponding TCP ACK packet to be forwarded from IG to A. The round-trip time in the wireless part, which we denote as $RTT_{wireless}$, is calculated as follows: Inspecting the transport layer TCP header, the Internet gateway IG maintains the packet sequence numbers of each TCP flow running between the wireless part and the wired part. When a TCP data packet with an arbitrary sequence number x is transmitted by A and reaches IG, the packet is forwarded to the wired destination and the forwarding time of the packet is recorded in a variable *T1* at IG. When the packet reaches the TCP destination B, gets acknowledged, and the corresponding TCP ACK arrives at IG, the arrival time of the TCP ACK packet is recorded in *T2*. The time difference between *T1* and *T2* is calculated and saved in the variable RTT_{wired}, which describes the packet RTT in the wired domain. Subsequently, IG writes RTT_{wired} into the *options field* of the TCP ACK header and forwards it towards A. When A receives the TCP ACK packet, it reads RTT_{wired} from the header and subtracts its value from the conventional RTT value, getting $RTT_{wireless}$ as a final result. Afterwards, the TCP sender uses $RTT_{wireless}$ to compute *FHD* and cov_{RTT} with respect to the wireless domain. For ease of exposition in (3) to (5), we assume that all wireless devices have the same bandwidth b.

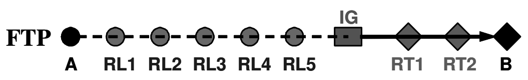


Figure 2. Wireless chain topology where A constitutes the wireless TCP source and B constitutes the wired TCP destination

Given that

$$t_q = \frac{1}{2}\left(\frac{RTT_{wireless}}{h} - \frac{s_{data} + s_{ACK}}{b} \right) \qquad (3)$$

we get:

$$FHD = 4\left(t_q + \frac{s_{data}}{b} \right) = 2\left(\frac{RTT_{wireless}}{h} + \frac{s_{data} - s_{ACK}}{b} \right) \qquad (4)$$

$$cov_{RTT} = \frac{\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(RTT^{i}_{wireless} - \overline{RTT}_{wireless} \right)^2}}{\overline{RTT}_{wireless}} \qquad (5)$$

Table 1 summarizes the parameters for the Gateway Adaptive Pacing scheme and their meaning. Note that h in Eqs. (3) and (4) denotes the number of hops between A and IG, which can be acquired from the routing protocol at IG. After computing *FHD* and cov_{RTT}, the adaptive transmission rate is computed as given in Eq. (1).

If wireless devices possess different bandwidths $b_1, b_2, \ldots, b_n$ (i.e. in a multi-rate mesh network), the individual bandwidths from intermediate devices along the current path are needed. Such information may be piggybacked to some TCP packets to prevent extra control overhead. Subsequently, *FHD* can be determined by considering the bandwidths $b_1, b_2, \ldots, b_n$ of individual links rather than the overall bandwidth b and appropriate summations.

4.3 Gateway Adaptive Pacing for Wired-to-Wireless Flows

To prevent any modifications of TCP in the wired domain, we choose to implement adaptive pacing on the Internet gateway, keeping the entire procedure hidden to the TCP source in the wired domain. Since the Internet gateway is essentially a network router, the adaptive pacing scheme is implemented on the IP layer. However, our approach is independent from the routing protocol employed in the wireless ad hoc extension of the Internet or the wireless mesh network.

Figure 3 illustrates the Gateway Adaptive Pacing procedure at the Internet gateway. TCP data packets are received from the wired domain through the wired interface and buffered in a FIFO queue which we denote as the pacing queue. Packets buffered in the pacing queue are then dequeued and transmitted rate-based through the wireless interface according to the current transmission rate, which is computed using Eqs. (1) to (5). Note that in the current case, h in Eqs. (3) and (4) denotes the number of hops between the Internet gateway and the wireless node constituting the TCP destination. $RTT_{wireless}$ describes the time taken for a TCP data packet to be forwarded from the Internet gateway to the wireless TCP destination plus the time taken for the corresponding TCP ACK to be forwarded from the wireless TCP destination to the Internet gateway. $RTT_{wireless}$ is computed in a similar way as in Section 4.2. That is, the transmission time of a TCP data packet x at the Internet gateway is recorded in a variable *T1*, whereas the arrival time of the corresponding TCP ACK at the Internet gateway is recorded in a

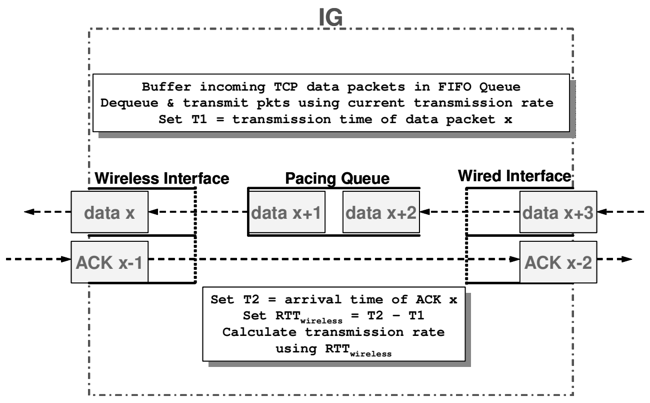
Table 1. Parameters for the Gateway Adaptive Pacing Scheme

Parameter	Meaning
h	Number of hops in wireless domain
b	Bandwidth of the wireless interface
$awnd_i$	Size of receiver advertized window for flow i
t_q	Average packet queuing delay per wireless node
s_{data}	Size of TCP data packet
s_{ACK}	Size of TCP ACK packet
RTT	Entire round trip time of TCP packets
$RTT_{wireless}$	Round trip time of TCP packets in wireless domain
RTT_{wired}	Round trip time of TCP packets in wired domain
cov_{RTT}	Coefficient of variation of RTT samples
FHD	Current 4-hop propagation delay in wireless domain
$\widehat{FHD}$	Exponentially weighted moving average of *FHD*

variable *T2*. Subsequently, $RTT_{wireless}$ is calculated by simply subtracting *T1* from *T2*.

Such rate-based packet transmission has the advantage of accounting for the deficiencies of IEEE 802.11, and thus improving the overall performance of TCP flows crossing both the wireless and wired domains. According to the number of wireless hops as well as the current contention in the wireless domain, the transmission rate can be adjusted for each flow separately in order to account for the different environment-specific influences experienced by each single flow. This is achieved by maintaining a flow-specific data structure at the Internet gateway which maintains the specific variables for each flow separately, e.g. packet sequence numbers, $RTT_{wireless}$, h, *FHD* as well as the current transmission rate *R*. Such flow-specific consideration assigns each TCP flow running through the Internet gateway a specific transmission rate, dependent on the contention experienced by this flow. In Section 5 we will show that such approach yields a significant performance improvement of both TCP goodput and fairness.

Utilizing our approach, TCP flows can be uniquely identified at the Internet gateway using the IP addresses and the port numbers of the TCP source and destination nodes. Note that for each TCP flow i, the Internet gateway has to provide at maximum a total of $awnd_i$ free packet buffer space in the pacing queue, where $awnd_i$ denotes the size of the receiver advertised window of flow i. That is, the wired TCP source of flow i can never transmit more than $awnd_i$ packets back to back before waiting for a corresponding TCP ACK to arrive.



IG
Buffer incoming TCP data packets in FIFO Queue
Dequeue & transmit pkts using current transmission rate
Set T1 = transmission time of data packet x
Wireless Interface
Pacing Queue
Wired Interface
data x
data x+1
data x+2
data x+3
ACK x-1
ACK x-2
Set T2 = arrival time of ACK x
Set RTT_wireless = T2 - T1
Calculate transmission rate
using RTT_wireless

Figure 3. The adaptive pacing procedure at the Internet gateway

Hence, we only need a constant number of free buffer space which equals awndi for a flow i. Accordingly, the total buffer space which should be provided at the Internet gateway is given by $\sum_{i=1}^{n} awnd_i$ packets, where n denotes the number of active TCP flows running through the Internet gateway. That is, for a flow with a TCP data packet size of 1460 bytes and a receiver advertised window of 64 packets, we only need 187 Kbytes of buffer space for caching packets at the Internet gateway, which make up about 1.4 Mbytes for 15 flows. In order to avoid unnecessary buffer space occupation, we also define two cases for deleting flow-specific queues and information at the Internet gateway. The first case is if the Internet gateway identifies a proper flow termination using the FIN-ACK sequence by the TCP entities, whereas the second case is if a certain timeout expires without receiving any packets for a given flow. Such timeout interval can be set to a few minutes. In the unlikely case that the buffer at the Internet gateway is completely occupied, pacing would be disabled for new flows until old flows terminate.

Note that the adaptive pacing algorithm is not affected if the delayed ACK option is used by the TCP receiver. The sole difference is that the new pacing rate gets computed less frequent since only every second TCP packet gets acknowledged by the TCP receiver. In fact, the adaptive pacing scheme combined with the delayed ACK option can significantly improve the goodput of TCP, as shown in [9].

4.4 Achieving Fairness for Oncoming Flows

As we will show in Section 5, applying adaptive pacing on the Internet gateway yields nearly optimal fairness between competing TCP flows in all scenarios without oncoming flows. However, in scenarios with two or more oncoming TCP flows where both wired-to-wireless as well as wireless-to-wired TCP flows pass through the Internet gateway, optimal fairness is not achieved. Consider for example the network topology depicted in Figure 4. Here, two parallel chains consisting of wireless nodes are connected to the Internet by the Internet gateway IG. The transmission range of each wireless node is 250m whereas both the interference range as well as the carrier sensing range are 550m. The distance between both chains is 400m. Thus, wireless nodes of opposite chains are within each other's interference range but out of each other's transmission range. Suppose there are two FTP transfers, the first (FTP1) running from the wired node B as FTP source to the wireless node A1 as FTP destination and the second (FTP2) running from the wireless node A2 as FTP source to the wired node B as FTP destination.

Simulation results for this scenario presented in Section 5 show that applying adaptive pacing on the Internet gateway significantly improves TCP fairness compared to standard TCP NewReno. However, FTP1 still achieves more goodput than FTP2. In order to get deeper insight, we analyze the TCP packet drop rate on link layer in the wireless domain. That is, we compute the number of TCP packets (data and ACKs) dropped at each wireless link in order to get insight on the state of the wireless link at each node. Table 2 shows the results of this study. It is conspicuous that the link RL7→RL8 on the lower chain experiences about 12 times more packet drops than the link RL3→RL4 which has the same relative position on the opposing upper chain. The same effect can be observed for the link RL8→IG on the lower chain which suffers about 2.5 times more packet drops than the corresponding link RL4→IG at the opposing upper chain. This explains why FTP 2, which runs on the lower chain, achieves less goodput than FTP 1. As we will explain in the subsequent discussion, the higher drop rate on the links RL7→RL8 and RL8→IG compared to the links RL3→RL4 and RL4→RL8 mainly depends on the interaction between two effects, namely the

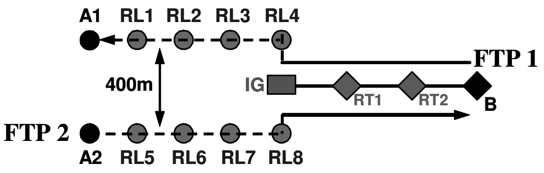


Figure 4. The two parallel chains topology

Table 2. Link layer packet drops for each wireless link in 1000 seconds simulation time

TCP ACKs	**A1→RL1**	**RL1→RL2**	**RL2→RL3**	**RL3→RL4**	**RL4→IG**
	8	146	112	12	40
TCP data	**A2→RL5**	**RL5→RL6**	**RL6→RL7**	**RL7→RL8**	**RL8→IG**
	11	128	147	141	102
TCP data	**RL1→A1**	**RL2→RL1**	**RL3→RL2**	**RL4→RL3**	**IG→RL4**
	1	54	90	8	40
TCP ACKs	**RL5→A2**	**RL6→RL5**	**RL7→RL6**	**RL8→RL7**	**IG→RL8**
	7	118	127	50	14

different packet sizes of TCP data and TCP ACK packets and the opposite directions of the flows.

Suppose RL7 wants to transmit a TCP data packet to RL8. Prior to the actual data transmission, RL7 and RL8 conduct an RTS/CTS handshake to avoid collisions with other transmissions. In case IG is concurrently transmitting TCP data packets to RL4 at the same time, then IG may constitute an exposed terminal for the transmission from RL7 to RL8, since RL8 hears the RTS packets from IG and thus does not respond with a CTS to RL7's RTS packets. After seven unsuccessful RTS attempts, RL7 drops the TCP data packet at link layer. In an analogous situation where RL3 wishes to transmit a TCP ACK packet to RL4, IG may constitute an exposed terminal for this transmission in case IG is concurrently transmitting TCP ACK packets to RL8. However, the difference between these two situations is that TCP data packets are much larger than TCP ACK packets and thus the probability for seven unsuccessful attempts for a RTS/CTS handshake is by far smaller than the case with TCP data packets. Furthermore, the loss of an ACK packet degrades TCP goodput less than the loss of a data packet, since TCP ACKs are cumulative, i.e. individual losses of ACKs can be overcome without retransmissions.

A further cause for the different goodput of the two TCP flows can be seen considering the transmission of TCP data packets from RL3 to RL2. These transmissions cause hidden terminal collisions at the receiving IG node, specifically for the transmission from RL8 to IG. That is, in case RL8 is transmitting a TCP data packet to IG at the same time when RL3 is transmitting a TCP data packet to RL2, the transmission from RL8 to IG will be corrupted whereas the transmission from RL3 to RL2 will succeed since RL2 lies beyond the interference range of RL8. Given that both transmissions incorporate large TCP data packets with relatively large transmission times, these collisions have a relatively high probability. In the analogous setting on the opposite chain, the transmission of TCP ACK packets from RL7 to RL6 can cause a collision at IG if RL4 is concurrently transmitting TCP ACK packets to IG at the same time. However, due to the reasons stated above, these collisions are less probable and thus cause less performance degradation than in the case of TCP data packets. In summary, due to the different flow directions and the different TCP packet sizes, FTP 1 takes advantage over FTP 2, resulting in less goodput and non-optimal fairness.

To solve this fairness problem, we extend the transport layer functionality added to the IP layer of the Internet gateway by incorporating *Goodput Control* for all TCP flows passing the Internet gateway. Goodput Control monitors the goodput of all TCP flows passing through the gateway and aims at achieving optimal fairness by throttling aggressive wired-to-wireless flows. That is, in case the Internet gateway IG recognizes that the goodput ratio between the goodput of a wired-to-wireless flow and the mean of the goodput of all flows exceeds a certain threshold *S*, then IG periodically probes the ability of the slower TCP flows to increase their goodput by throttling the rate of the faster TCP flows down to the value of the mean goodput. Note that since wired-to-wireless flows gain more goodput than wireless-to-wired flows, this throttling can easily be performed by adjusting the transmission rate of the Gateway Adaptive Pacing algorithm. Throttling the fast TCP flows may result in either:

(1) an increase of the goodput achieved by the slower flows in case they contend with the fast flows, or
(2) no change in the goodput of the slower flows in case there is no contention.

Considering case (1), the throttling is effective for improving TCP fairness between competing flows, while in case (2), throttling fast flows would not yield any benefit for slow flows, but would rather unnecessarily decrease the goodput of the fast flows. Thus, in case (2), the throttling is disabled. This way, fast flows are only throttled in case they affect the goodput of slow flows, i.e. in case both fast and slow flows share the same bottleneck. To maintain the responsiveness of our approach to changing network conditions, we continuously verify whether throttling is still necessary. This is done by applying an aging algorithm to the throttling value, i.e., with increasing time the degree of throttling decreases in order to account to changing traffic conditions after which the throttling might be unnecessary. Furthermore, whenever IG recognizes a termination of a TCP flow, it resets all throttling-specific variables. In case the unfairness still remains, it is handled during the next periodic probing. As verified by our simulations, suitable values for the throttling parameters are 5 seconds for the throttling interval as well as 1.1 for the threshold *S*, i.e., a fast TCP flow may at maximum achieve 10% more goodput than then mean goodput of all flows, or else it gets temporarily throttled.

As we will show in Section 5, using this Goodput Control algorithm, the fairness of competing TCP flows can be optimized while avoiding any additional control traffic overhead or requiring global knowledge about the network topology. Recall that our approach is implemented at the Internet gateway only, which is not affected by energy consumption issues and has sufficient processing power and memory. Consider that the Goodput Control approach only works for TCP flows passing the same Internet gateway. Nevertheless, there might be network topologies in which similar effects as described above cause unfairness between TCP flows passing different Internet gateways. However, we argue that in multihop extensions to the Internet or mesh networks, these scenarios are rare since Internet gateways are typically located in substantial distances. Otherwise, single-hop wireless Internet access would rather be deployed than multihop wireless extensions of the Internet. Thus, our solution is beneficial in almost all considered scenarios. In the remainder of this paper, we denote our Gateway Adaptive Pacing scheme including Goodput Control as *TCP with Gateway Adaptive Pacing (TCP-GAP).*

4.5 Dealing with Handovers due to Mobility

In wireless mesh networks, there may be scenarios where the TCP entities in the wireless domain constitute mobile devices which move along multiple gateways. Due to the mobility in such scenarios, a handover procedure has to be performed between the Internet gateways by the routing layer. That is, as the mobile device moves, it may find Internet gateways to which it has a shorter route than the current Internet gateway.

The advantage of TCP-GAP in such scenarios is that it does not require any exchanging of hard-state information about the TCP connections between the Internet gateways. Using TCP-GAP, Internet gateways only maintain soft-state information about TCP connections which can be built up from scratch by new Internet gateways after a handover procedure. Other approaches such as the split connection approach [4] require complicated handover procedures between Internet gateways as they maintain hard-state information about TCP connections, which have to be transferred to new Internet gateways.

In Section 5, we consider scenarios with a single Internet gateway and no mobility. In a further simulation study not included due to space limitations, we consider the case where the TCP entities in the wireless domain constitute mobile devices which move along multiple gateways with a pedestrian speed of max. 2 m/s. The study shows that the handover procedure between multiple Internet gateways does not affect the performance improvements gained by TCP-GAP.

4.6 Pseudo Code for TCP-GAP

To provide intuition on how to implement TCP-GAP, we provide pseudo code for the Gateway Adaptive Pacing scheme as well as for the Goodput Control approach. The implementation involves the wireless TCP sender as well as the Internet gateway. Figure 5 outlines the functionality added to the TCP implementation at the wireless TCP sender, whereas Figure 6 shows the functionality which has to be added to the IP layer implementation at the Internet gateway. Recall that these additions are independent of the applied routing protocol as long as the number of hops to wireless nodes and the bandwidth of the wireless interface are provided.

```
proc recv_pkt() {
foreach received TCP ACK do
  read $RTT_{wired}$ from TCP header
  set $RTT_{wireless} = RTT - RTT_{wired}$
  calculate transmission rate using $RTT_{wireless}$
done
}
```

Figure 5. Pseudo code for TCP-GAP implemented at the wireless TCP sender

```
Global Variables (used to identify TCP flows):
  seq: pkt sequence number
  sip: source IP
  dip: destination IP
  sp: source port
  dp: destination port
proc recv_pkt() {
  if (packet type == TCP data) then
    read [seq, sip, dip, sp, dp] out of packet header
    switch (direction) {
      case (wireless to wired):
        set $T1^{[seq, sip,dip,sp,dp]}$ = current time
        forward packet
      case (wired to wireless):
        buffer packet in Pacing Queue
        (no processing here, packets dequeued later)
    }
  else if (packet type == TCP ACK) then
    read [seq, sip, dip, sp, dp] out of packet header
    switch (direction) {
      case (wireless to wired):
        if (duplicated ACK) then
          retransmit data pkt with sequence number
          (seq+1)
        else
          set $RTT_{wireless}^{[seq, sip, dip, sp, dp]} = T2^{[seq, sip,dip,sp,dp]} - T1^{[seq, sip,dip,sp,dp]}$
          update transmission rate using
          $RTT_{wireless}^{[seq, sip, dip, sp, dp]}$
```

```
            forward packet
        endif
      case (wired to wireless):
        set $T2^{[seq, sip,dip,sp,dp]}$ = current time
        set $RTT_{wired}^{[seq, sip, dip, sp, dp]} = T2^{[seq, sip,dip,sp,dp]} - T1^{[seq, sip,dip,sp,dp]}$
        write $RTT_{wired}^{[seq, sip, dip, sp, dp]}$ into TCP header of packet
        forward packet
      }
  endif
proc dequeue_pacing_queue() {
comment: procedure called upon expiration of rate-based timer to dequeue pacing queue
  dequeue TCP data packet (FIFO order)
  set $T1^{[seq, sip,dip,sp,dp]}$ = current time
  transmit packet through wireless interface
}
proc goodput_control() {
  Local Variables:
    interval: Probing interval (set to 5s in simulations)
    G(i): Goodput achieved by a flow i
    $G_{avg}$: Average goodput of all flows
    S: Threshold which defines acceptable goodput
       deviation between oncoming flows (set to 1.1
       in simulations)
    R(i): Current transmission rate of flow i
    once every interval seconds do
      foreach flow i do
        set deviation(i) = $G(i)/G_{avg}$
        if (deviation(i) > S and i is wired-to-wireless)
        then
          throttle rate of flow i to $G_{avg}$
        else if (throttling is on and no improvement for
               slow flows)
        then
          cancel throttling of flow i
        endif
      done
      if (a connection terminates or starts) then
        reset all throttle-specific variables
        cancel throttling of all flows
      endif
      perform aging by increasing rate of flow i to
      R(i) + 1.1 * deviation(i)
    done
}
```

Figure 6. Pseudo code for TCP-GAP implemented at the Internet gateway

5 PERFORMANCE EVALUATION

5.1 Simulation Environment

The simulation experiments in this paper are conducted using the network simulator ns-2 [10]. In the wireless domain, the MAC layer parameters of IEEE 802.11 are configured to provide a transmission range of 250m and a carrier sensing range as well as an interference range of 550m, as consistent with a Lucent WaveLan DSSS radio interface. The transmission of each data packet on the MAC layer is preceded by a Request-To-Send/Clear-To-Send (RTS/CTS) handshake. We consider a wireless channel bandwidth of 11 Mbit/s as supported by IEEE 802.11b and set the size of TCP data packets to 1460 bytes. Unless otherwise stated, in the wireless domain of all considered topologies, each node is 200 meters apart from each of its adjacent nodes. As ad hoc routing protocol for packet routing in the wireless domain we use AODV [14]. Unless otherwise stated, we set the bandwidth of the full-duplex wired links to 10 Mbit/s and the packet delay to 40ms.

In all experiments, except for experiments showing transient behavior, we conduct steady-state simulations starting with an initially idle system. In each run, we simulate TCP flows until 55.000 packets are successfully transmitted, and split the simulation output in 11 batches of size 5.000 packets. The first batch is discarded as initial transient. The considered performance measures are derived from the remaining 10 batches with 95% confidence intervals by the batch means method.

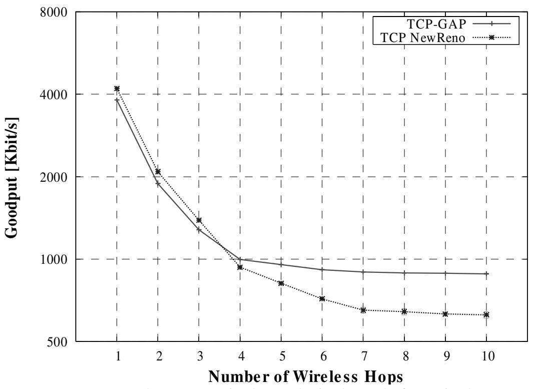


Figure 7. TCP goodput vs. wireless chain length for wireless-to-wired flows

5.2 Chain Topology

First we consider a chain topology as depicted in Figure 2 of Section 4.2. In the first experiment, we define an FTP flow running from the wireless node A to the wired host B, where we vary the length of the wireless router-chain and plot the achieved goodput accordingly. Figure 7 shows the goodput versus number of wireless hops h of TCP-GAP as well as TCP NewReno. We observe that for $h < 4$ where no hidden terminals are present, TCP NewReno achieves slightly higher goodput than TCP-GAP. That is, the bursty transmission of TCP NewReno gains a slight advantage over the adaptive pacing of TCP-GAP, since the IEEE 802.11 MAC scheduling prevents packet losses caused by hidden terminals for less than four hops. However, in our simulation, we noticed that the bursty traffic of TCP NewReno results in severe unfairness in scenarios with multiple flows, even in topologies where no hidden terminal is present. Therefore, instead of disabling the adaptive pacing scheme for wireless routes with less than four hops, TCP-GAP computes the transmission rate using the h-hop delay and achieves best fairness results due to its adaptive pacing scheme. For chains with $h \geq 4$, we observe that TCP-GAP achieves up to 41% more goodput than TCP NewReno due to the presence of the hidden terminal problem. Such performance improvement is the result of the consideration of the IEEE 802.11 spatial reuse constraint in the computation of the TCP-GAP adaptive pacing rate.

In the second experiment, we consider the opposite case where the wired host B constitutes the TCP sender and the wireless node A constitutes the TCP destination. Figure 8 shows that for $h < 4$, both TCP variants achieve similar goodput with a maximum of 3% value deviation. For $h \geq 4$, the adaptive rate-based transmission of TCP-GAP effectively decreases network congestion and achieves up to 42% more goodput than TCP NewReno.

5.3 Parallel Chains Topology

As a second topology, we consider two parallel chains as shown in Figure 4 of Section 4.4. Consistent with the previous scenario, we define three wired nodes while we set two FTP flows running between the wireless and wired domains. We consider three different traffic scenarios, where each case corresponds to a specific adjustment of the flow directions. That is, first we consider the case where both FTP flows start at the nodes A1 and A2 as TCP sources and end at the wired host B as TCP destination. Second we consider the opposite direction where both FTP flows start at B and end at A1 and A2, respectively. In the final scenario we examine mixed flow directions where FTP 1 runs from A1 as source to B as destination and FTP 2 runs from B as source to A2 as destination. Figures 9 to

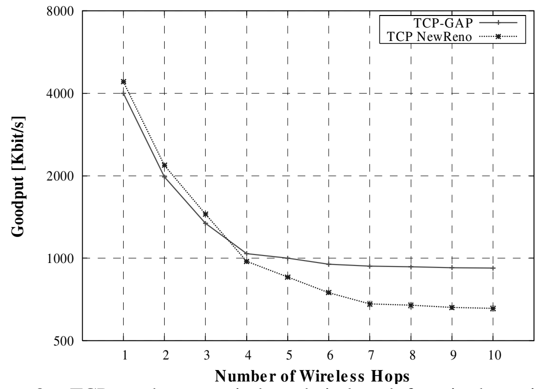


Figure 8. TCP goodput vs. wireless chain length for wired-to-wireless flows

11 show the results of this simulation, where each figure corresponds to a specific adjustment of the flow directions. The figures plot the individual goodput of each FTP flow as well as the aggregate goodput, which is defined as the sum of the goodput achieved by both flows.

In Figures 9 and 10 we see that TCP-GAP significantly outperforms TCP NewReno both in terms of fairness and goodput. Using TCP NewReno, FTP 1 occupies the entire available bandwidth at cost of FTP 2, while both flows share the available bandwidth equally using TCP-GAP. In fact, TCP-GAP also achieves a higher aggregate goodput than TCP NewReno. Note that approaches which aim to improve TCP performance by exchanging control information between wireless nodes would not work in such scenarios since no direct communication is possible between nodes belonging to one chain and nodes belonging to the opposite chain due to the 400m inter-chain distance.

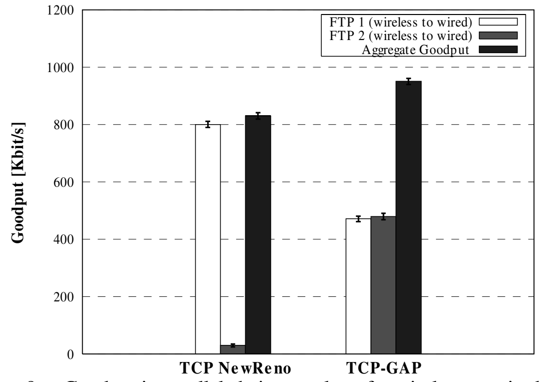


Figure 9. Goodput in parallel chains topology for wireless-to-wired flows

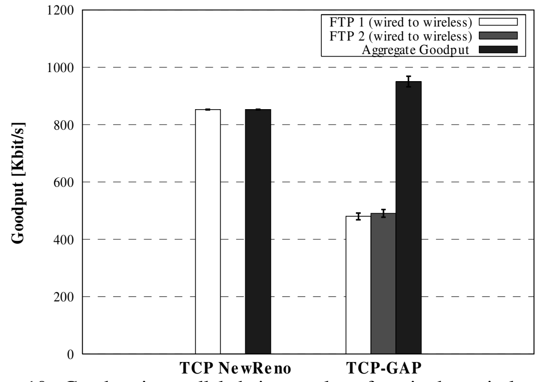


Figure 10. Goodput in parallel chains topology for wired-to-wireless flows

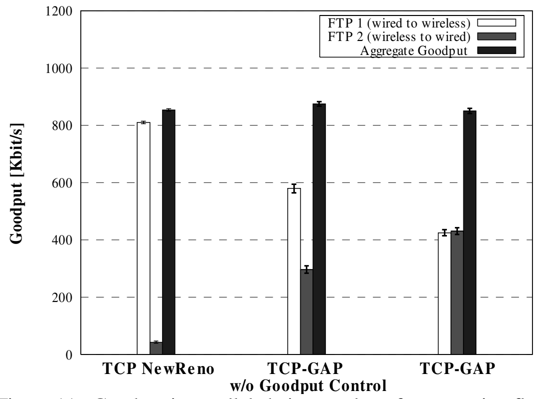


Figure 11. Goodput in parallel chains topology for oncoming flows

Figure 11 plots the results for the case with mixed flow direction where FTP 1 runs from B to A1 and FTP 2 runs from A2 to B. We observe that TCP-GAP without Goodput Control achieves much better fairness than TCP NewReno, although the fairness is not optimal like in the previous two cases. Such unfairness between oncoming flows was also observed in [16] and [18] and further discussed in Section 4. In Figure 11 we observe that, due to the Goodput Control scheme, TCP-GAP achieves optimal fairness with almost no sacrifice of the aggregate goodput. This shows that the Goodput Control scheme constitutes an effective method for achieving optimal fairness between oncoming flows.

Responsiveness

In order to evaluate how quickly a specific TCP variant responds to changing traffic conditions in the network, we conduct a further simulation using the parallel chains topology. We define two FTP flows which run from the wired domain to the wireless domain, i.e. FTP 1 starts at B and ends at A1 whereas FTP 2 starts at B and ends at A2. While FTP 1 runs from the beginning of the simulation until the end, FTP 2 runs from the beginning of the simulation and stops at time N1=130s, then restarts again at time N2=160s where it continues until the end. We are interested in studying how FTP 1 reacts upon the stopping and starting of FTP 2. Figures 12 and 13 plot the goodput of both flows versus simulation time for TCP-GAP and TCP NewReno, respectively. Considering TCP-GAP, we observe that FTP 1 quickly takes advantage of the entire available bandwidth when FTP 2 stops, while both flows share the bandwidth fairly when they content for the channel. As for TCP NewReno, we see that FTP 1 occupies the entire available bandwidth at cost of FTP 2, which completely starves. We conclude that TCP-GAP not only provides superior fairness compared to TCP NewReno but also quickly responds to changing network conditions.

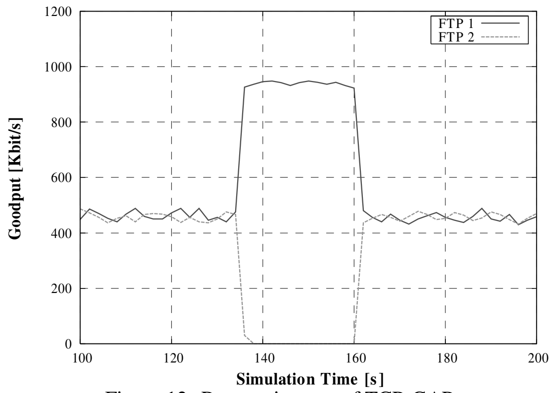


Figure 12. Responsiveness of TCP-GAP

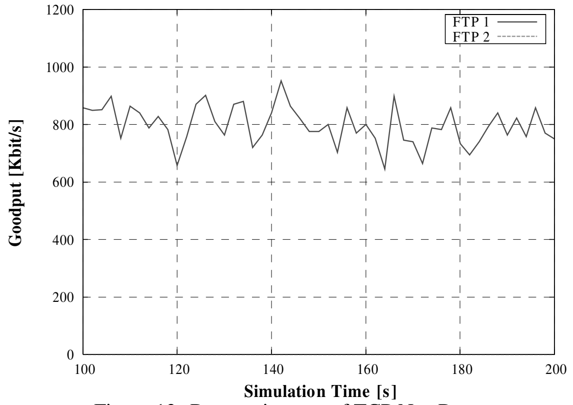


Figure 13. Responsiveness of TCP NewReno

In Section 5.3 we will show how this improved responsiveness results in substantial improvement in aggregate goodput for short TCP flows.

5.4 Cross Topology

As a third and more complex topology we consider a cross of wireless nodes, where the Internet gateway IG is positioned at the center of the cross as depicted in Figure 14. The wired domain comprises seven wired hosts, which are depicted as diamonds. We define four FTP flows and consider similar flow directions as for the previous topology. That is, in case (1), which is depicted in Figure 14, all FTP flows run from the wireless to the wired domain with the TCP source and destination entities (A1→B1), (A2→B2), (A3→B3) and (A4→B4). In case (2), we consider the opposite direction where the flows start in the wired and end in the wireless domain, where the TCP entities are given by (B1→A1), (B2→A2), (B3→A3) and (B4→A4). Finally, we consider the mixed case where two FTP flows run from the wireless to the wired domain and the other two flows run the other way round, given the TCP entities (A1→B1), (B2→A2), (A3→B3) and (B4→A4).

Figures 15 to 17 show the results of this simulation. Consistent with the previous results, the figures show that TCP-GAP considerably outperforms TCP NewReno both in terms of fairness and aggregate goodput. In fact, TCP-GAP achieves optimal fairness between the competing flows in the first two cases. Consistent with the mixed case of the previous simulation, in Figure 17, we notice that for TCP-GAP without Goodput Control, the first two flows get slightly less goodput than the other two flows. From this figure we conclude that using Goodput Control yields optimal fairness between oncoming flows, even in cases where we have multiple wireless-to-wired as well as wired-to-wireless flows.

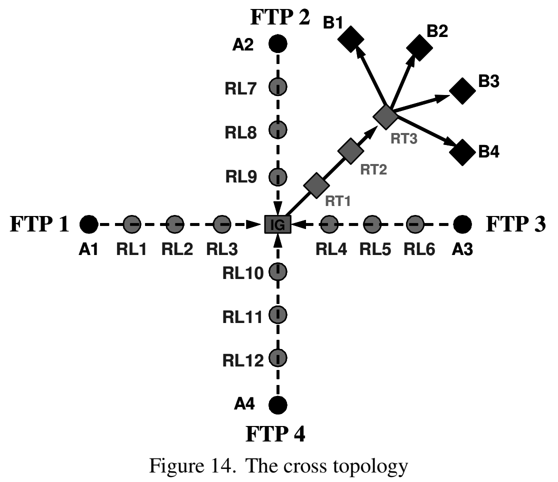


Figure 14. The cross topology

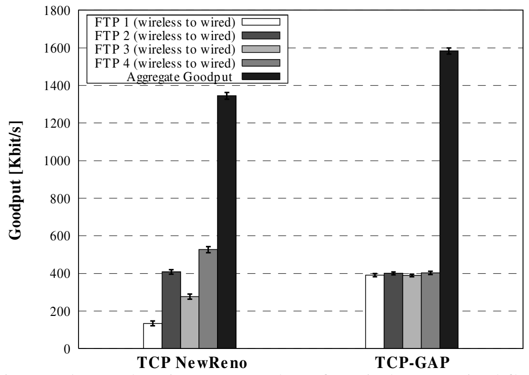


Figure 15. Goodput in cross topology for wireless-to-wired flows

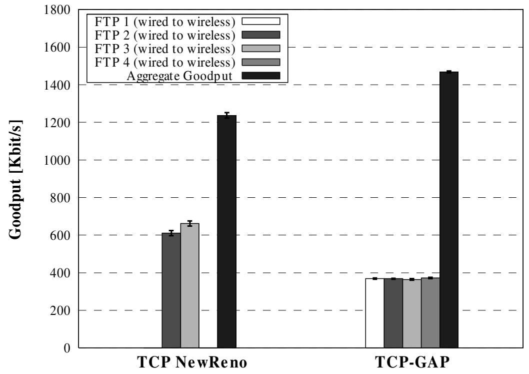


Figure 16. Goodput in cross topology for wired-to-wireless flows

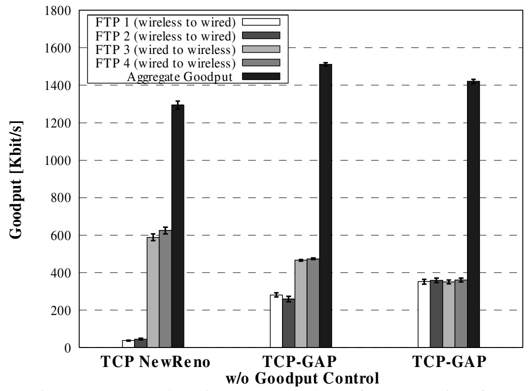


Figure 17. Goodput in cross topology for oncoming flow

5.5 Random Topology

As a final topology we consider a random topology of 120 wireless nodes uniformly distributed on a flat area of 2500m x 1000m. According to [5], all nodes in the wireless domain can communicate with each other over one or more hops with probability *P*=99.9%. Similar networks do already exist such as the MIT Roofnet which builds up an unplanned IEEE 802.11b wireless mesh network over an urban area of about four square kilometers [6]. We define eight FTP flows with randomly chosen TCP source and destination pairs, where FTP 1 to FTP 4 run from the wired to the wireless domain and FTP 5 to FTP 8 run in the opposite direction. The position of the Internet gateway is also randomly selected while we define two routers and one host in the wired domain similar to the wired nodes depicted in Figure 4. Thereby, the wired host B3 constitutes the TCP source for FTP 1 to FTP 4 and the TCP

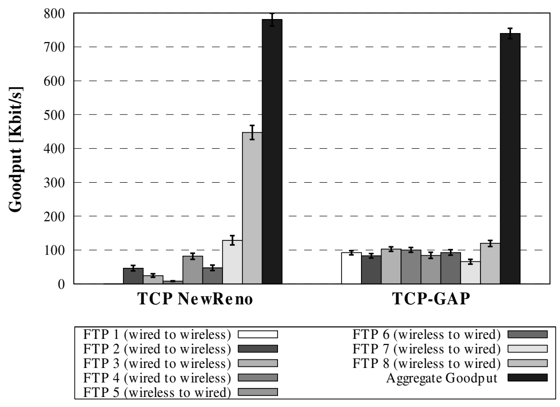


Figure 18. Goodput in random topology for oncoming flows running on paths of different lengths

destination for FTP 5 to FTP 8. Opposed to previous experiments, in this simulation TCP flows run on paths of different lengths.

Figure 18 shows that TCP-GAP achieves much better fairness between the flows than TCP NewReno. Specifically, TCP NewReno lets FTP 1 and FTP 4 almost completely starve while all flows get a fraction of the available bandwidth using TCP-GAP. We notice that TCP-GAP achieves slightly less aggregate goodput than TCP NewReno due to the well known tradeoff between aggregate goodput and fairness which is caused by the absence of optimal scheduling of IEEE 802.11.

This problem is further discussed in [17]. Note that the different wireless path lengths of the considered flows may have further impact on the fairness between multiple flows. Such effects are not further investigated in this paper and are subject to future work.

6 CONCLUSION

In this paper, we proposed an adaptive pacing scheme on the Internet gateway for improving both goodput and fairness of TCP flows in hybrid wireless/wired networks. Our approach, denoted as TCP with Gateway Adaptive Pacing (TCP-GAP), accounts for the different characteristics of the wireless and wired domains by deploying rate-based congestion control for the wireless part of the network at the Internet gateway. Furthermore, we gave insight on the reasons of the unfairness in case of oncoming flows where both wired-to-wireless as well as wireless-to-wired connections pass through the Internet gateway. Subsequently, we introduced a goodput control scheme at the Internet gateway in order to achieve nearly optimal fairness for such scenarios.

We showed that nearly optimal fairness between multiple TCP flows in hybrid wireless/wired networks can be achieved by solely modifying the transport layer. Thus, opposed to [12] and [18], TCP-GAP is easily deployable since it does neither require any modifications of standard TCP in the wired domain nor modifications on the link or network layers.

In future work, we are measuring the performance of TCP-GAP in a real testbed.

7 REFERENCES

[1] A. Aggrawal, S. Savage, and T. Anderson, Understanding the Performance of TCP Pacing, Proc. IEEE INFOCOM, Tel Aviv, Israel, 2000.

[2] Ö. B. Akan and I. F. Akyildiz, ATL: An Adaptive Transport Layer Suite for Next-Generation Wireless Internet, IEEE Journal on Selected Areas in Communications, 22, 2004.

[3] I. F. Akyildiz, X. Wang, and W. Wang, Wireless Mesh Networks: a survey, Computer Networks, 47, 2005.

[4] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, A Comparison of Mechanisms for Improving TCP Performance over Wireless Links, IEEE/ACM Transactions on Networking, 5, 1997.

[5] C. Bettstetter, On the Minimum Node Degree and Connectivity of a Wireless Multihop Network, Proc. ACM MOBIHOC, Lausanne, Switzerland, 2002.

[6] J. Bicket, D. Aguayo, S. Biswas, and R. Morris, Architecture and Evaluation of an Unplanned 802.11b Mesh Network, Proc. ACM MOBICOM, Cologne, Germany, 2005.

[7] D. De Couto, D. Aguayo, J. Bicket, and R. Morris, A High-Throughput Path Metric for Multi-Hop Wireless Routing, Proc. ACM MOBICOM, San Diego, CA, 2003.

[8] R. Draves, J. Padhye, and B. Zill, Routing in multi-radio, multi-hop wireless mesh networks, Proc. ACM MOBICOM, Philadelphia, PA, 2004.

[9] S. ElRakabawy, A. Klemm, and C. Lindemann, TCP with Adaptive Pacing for Multihop Wireless Networks, Proc. ACM MOBIHOC, Urbana-Champaign, IL, 2005.

[10] K. Fall and K. Varadhan (Ed.), The ns-2 Manual, Technical Report, The VINT Project, UC Berkeley, LBL, USC/ISI and Xerox PARC, 2005.

[11] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, The Impact of Multihop Wireless Channel on TCP Throughput and Loss, Proc. IEEE INFOCOM, San Francisco CA, 2003.

[12] V. Gambiroza, B. Sadeghi, and E. Knightly, End-to-End Performance and Fairness in Multihop Wireless Backhaul Networks, Proc. ACM MOBICOM, Philadelphia, PA, 2004.

[13] S. Mascolo, C. Casetti, M. Gerla, M. Sandidi, and R. Wang, TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links, Proc. ACM MOBICOM, Rome, Italy, 2001.

[14] C. Perkins, E. Royer, and S. Das, Ad hoc On-Demand Distance Vector (AODV) Routing, IETF RFC 3561, 2003.

[15] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar, ATP: A Reliable Transport Protocol for Ad Hoc Networks, Proc. ACM MOBIHOC, Annapolis, MA, 2003.

[16] K. Xu, S. Bae, S. Lee, and M. Gerla, TCP Behavior across Multihop Wireless Networks and the Wired Internet, Proc. ACM WoWMoM, Atlanta, GA, 2002.

[17] K. Xu, M. Gerla, L. Qi, and Y. Shu, Enhancing TCP Fairness in Ad Hoc Wireless Networks using Neighborhood RED, Proc. ACM MOBICOM, San Diego CA, 2003.

[18] L. Yang, W. Seah, and Q. Yin, Improving Fairness among TCP Flows crossing Wireless Ad Hoc and Wired Networks, Proc. ACM MOBIHOC, Annapolis MD, 2003.