Available online at www.sciencedirect.com

ScienceDirect

Computer Networks 52 (2008) 180–198

Computer Networks

www.elsevier.com/locate/comnet

TCP with gateway adaptive pacing for multihop wireless networks with Internet connectivity ☆

Sherif M. ElRakabawy, Alexander Klemm, Christoph Lindemann *

University of Leipzig, Department of Computer Science, Johannisgasse 26, 04103 Leipzig, Germany

Available online 29 September 2007

Abstract

This paper introduces an effective congestion control pacing scheme for TCP over multihop wireless networks with Internet connectivity. The pacing scheme is implemented at the wireless TCP sender as well as at the Internet gateway, and reacts according to the direction of TCP flows running across the wireless network and the Internet. Moreover, 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. The proposed scheme, which we denote as TCP with Gateway Adaptive Pacing (TCP-GAP), does not impose any control traffic overhead for achieving fairness among active TCP flows and can be incrementally deployed since it does not require any modifications of TCP in the wired part of the network. In an extensive set of experiments using ns-2 we 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 for FTP-like traffic as well as up to 70% more goodput for HTTP-like traffic than TCP NewReno. We also investigate the sensitivity of the considered TCP variants to different bandwidths of the wired and wireless links with respect to both aggregate goodput and fairness.

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

1. Introduction

In recent years, rapidly emerging wireless networks with Internet connectivity, such as wireless mesh networks [3], have been gaining increasing importance in academic research and industry. Common Internet applications such as Web browsing, e-mail and file transfer over such hybrid wireless/wired networks require TCP as the 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

☆ A preliminary version of this article was published in [9]. This work was funded in part by the German Research Council (DFG) under Grant Li-645/18-1.

* Corresponding author. Tel.: +49 341 97 32270; fax: +49 341 97 32289.

E-mail addresses: sme@informatik.uni-leipzig.de (S.M. ElRakabawy), ak@informatik.uni-leipzig.de (A. Klemm), cl@informatik.uni-leipzig.de (C. Lindemann).

1389-1286/$ - see front matter © 2007 Published by Elsevier B.V.
doi:10.1016/j.comnet.2007.09.016

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 [16] 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 [18]. 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. For the emergence of commercial applications in hybrid wireless/wired networks, more advanced approaches are needed which provide means for retaining the end-to-end semantics of TCP and require neither modifications at the network layer nor at the link layer of mobile 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 [8], 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 [17,19]. 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 as *TCP with Gateway Adaptive Pacing* (*TCP-GAP*). In contrast to previous work [12,18], 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.

We evaluate both the steady-state as well as the transient behavior of TCP-GAP using ns-2 simulation [10] with IEEE 802.11b/g, where we deploy scenarios describing different node topologies, different link-layer bandwidths as well as different traffic patterns. 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 for FTP-like traffic as well as up to 70% more goodput for HTTP-like traffic.

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 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 vs. TCP NewReno is presented in Section 5. Finally, concluding remarks are given.

2. Related work

Several TCP enhancements (e.g., [8,11,18]) and new transport protocols such as ATP [16] 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 [8], 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 [8], we propose an adaptive pacing scheme at the TCP sender. Contrary to [8], 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. [19] 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 [19] 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 [19], 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. The four-hop propagation delay 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. Deploying such an adaptive pacing scheme, 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 [19]. Beyond [19], 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 at the transport layer rather than modification at 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. Therefore; we consider max–min fairness of TCP flows rather than TAP-fairness.

Mascolo et al. [14] 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 and Akyildiz [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 [2,14], we consider hybrid wireless/wired networks, in which the wireless part comprises 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 network class

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 [15] or other routing protocols adopting ETX [7] or ETT [6] as its own routing protocol. Fig. 1 illustrates the considered class of wireless/wired networks. Note that the network architecture illustrated in Fig. 1 can be consid-

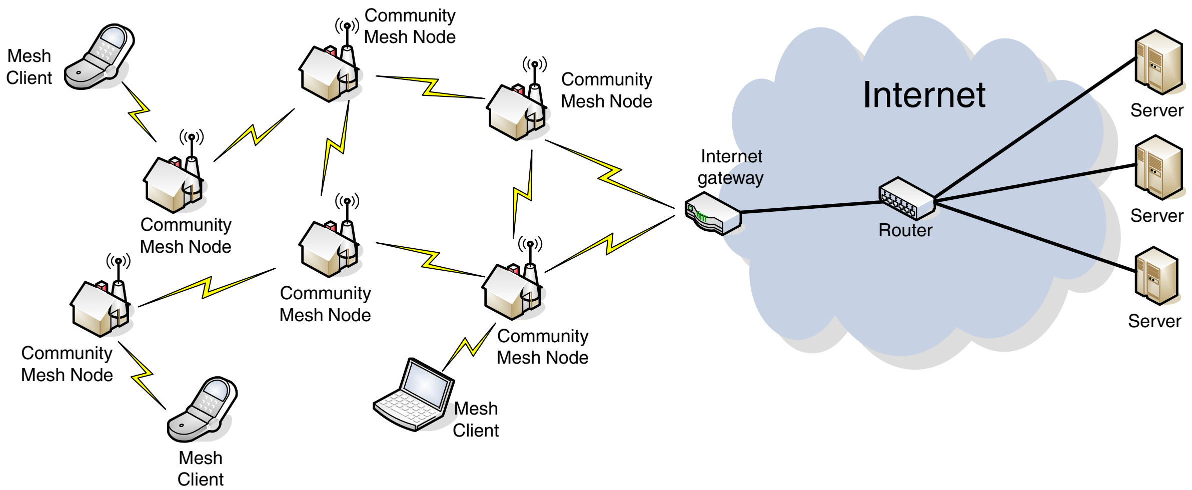

Fig. 1. Targeted network architecture: opportunistic ad hoc extension to the Internet and unplanned, single-radio wireless mesh networks.

ered 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 200 m. In fact, the hidden terminal problem can even occur to a larger extent in topologies with irregular node placement. This is due to the fact that the ideal case with inter-node distances of 200 m (given a wireless transmission range of 250 m) 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 lower bound for the number of hidden terminals. Nevertheless, we also consider random topologies with irregular node placement in order to verify the applicability of our approach in such environments.

Conventional ad hoc routing protocols such as AODV [15] use the minimum hop count as the routing metric. For wireless mesh networks, novel routing metrics like the expected transmission count (ETX) [7] and the expected transmission time (ETT) [6] 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 gateway adaptive pacing scheme

4.1. Dealing with the deficiencies of IEEE 802.11

In order to improve 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. We recall the spatial reuse constraint of IEEE 802.11 multihop networks, which is reported in previous studies such as [11], by considering the static chain topology depicted in Fig. 2 where the inter-node distance is

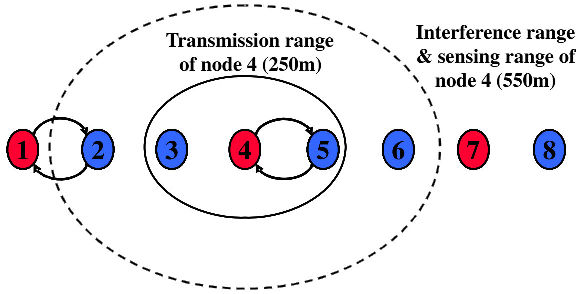

Fig. 2. A chain of nodes showing the hidden terminal effect.

200 m. Assume that node 1 wishes to transmit data to node 2 and node 4 wishes to transmit data to node 5. In this topology, node 4 is a hidden terminal for the transmission from node 1 to node 2. That is, node 4 can neither receive the RTS/CTS handshake between node 1 and node 2 nor sense the data transmission from node 1 to node 2, since node 1 is out of the sensing range of node 4. Thus, node 4 may transmit to node 5 while node 1 is transmitting to node 2. This causes a collision at the receiving node 2, since node 2 is within the interference range of node 4.

The spatial reuse constraint is accounted for 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 $\mathrm{cov}_{\mathrm{RTT}}$. In summary, the adaptive transmission rate R computed by the TCP sender is given by [8]

$$R = \frac{1}{\widehat{\mathrm{FHD}} \cdot (1 + 2\mathrm{cov}_{\mathrm{RTT}})}, \quad (1)$$

where

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

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 at the 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. Wireless-to-wired flows

We consider the case where a wireless node constitutes the TCP source and a host in the wired domain is the TCP destination. Fig. 3 illustrates such a 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 packets crossing both the wireless and the wired parts of the network. However, for deriving proper estimates for FHD and $\mathrm{cov}_{\mathrm{RTT}}$, 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 $\mathrm{RTT}_{\mathrm{wireless}}$, 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

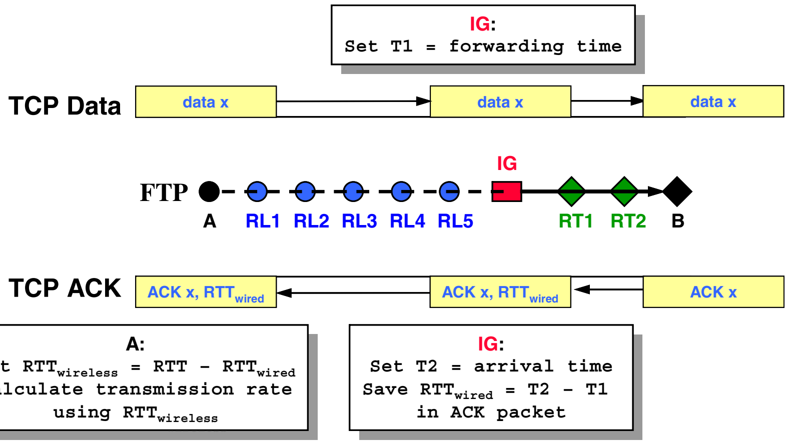

Fig. 3. Wireless chain topology where A constitutes the wireless TCP source and B constitutes the wired TCP destination.

in *T2*. The time difference between *T1* and *T2* is calculated and saved in the variable $\mathrm{RTT}_{\mathrm{wired}}$, which describes the packet RTT in the wired domain. Subsequently, IG writes $\mathrm{RTT}_{\mathrm{wired}}$ into the *options field* of the TCP ACK packet, which begins after the main 20-bytes TCP header, and forwards it towards A. $\mathrm{RTT}_{\mathrm{wired}}$ is recorded in terms of milliseconds, requiring only 16 bits of space in the options field to express values up to 65,535 ms. When A receives the TCP ACK packet, it reads RTT_{wired} from the header and subtracts its value from the conventional RTT value, getting $\mathrm{RTT}_{\mathrm{wireless}}$ as a final result. Afterwards, the TCP sender uses $\mathrm{RTT}_{\mathrm{wireless}}$ to compute FHD and $\mathrm{cov}_{\mathrm{RTT}}$ with respect to the wireless domain. For ease of exposition in (3)–(5), we assume that all wireless devices have the same bandwidth b.

Given that

$$t_q = \frac{1}{2}\left(\frac{\mathrm{RTT}_{\mathrm{wireless}}}{h} - \frac{s_{\mathrm{data}} + s_{\mathrm{ACK}}}{b}\right), \tag{3}$$

we get

$$\mathrm{FHD} = 4\left(t_q + \frac{s_{\mathrm{data}}}{b}\right) = 2\left(\frac{\mathrm{RTT}_{\mathrm{wireless}}}{h} + \frac{s_{\mathrm{data}} - s_{\mathrm{ACK}}}{b}\right), \tag{4}$$

$$\mathrm{cov}_{\mathrm{RTT}} = \frac{\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(\mathrm{RTT}^{i}_{\mathrm{wireless}} - \overline{\mathrm{RTT}}_{\mathrm{wireless}}\right)^2}}{\overline{\mathrm{RTT}}_{\mathrm{wireless}}}. \tag{5}$$

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 $\mathrm{cov}_{\mathrm{RTT}}$, the adaptive transmission rate is computed as given in Eq. (1).

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
$\mathrm{RTT}_{\mathrm{wireless}}$	Round trip time of TCP packets in wireless domain
$\mathrm{RTT}_{\mathrm{wired}}$	Round trip time of TCP packets in wired domain
$\mathrm{cov}_{\mathrm{RTT}}$	Coefficient of variation of RTT samples
FHD	Current four-hop propagation delay in wireless domain
$\widehat{\mathrm{FHD}}$	Exponentially weighted moving average of FHD

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 on 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.

In case the TCP stack at the wireless source nodes does not support the adaptive pacing scheme, the wireless TCP source would simply transmit packets according to the implemented TCP version (e.g., TCP NewReno), since the gateway adaptive pacing approach is backward compatible to standard TCP variants. Specifically, the Internet gateway would simply forward the TCP packets to the wired domain without affecting the operation of standard TCP.

4.3. Wired-to-wireless flows

As a main design goal of our approach is to prevent any modifications of TCP in the wired domain, we choose to implement adaptive pacing at 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 at 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.

Fig. 4 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)–(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 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.

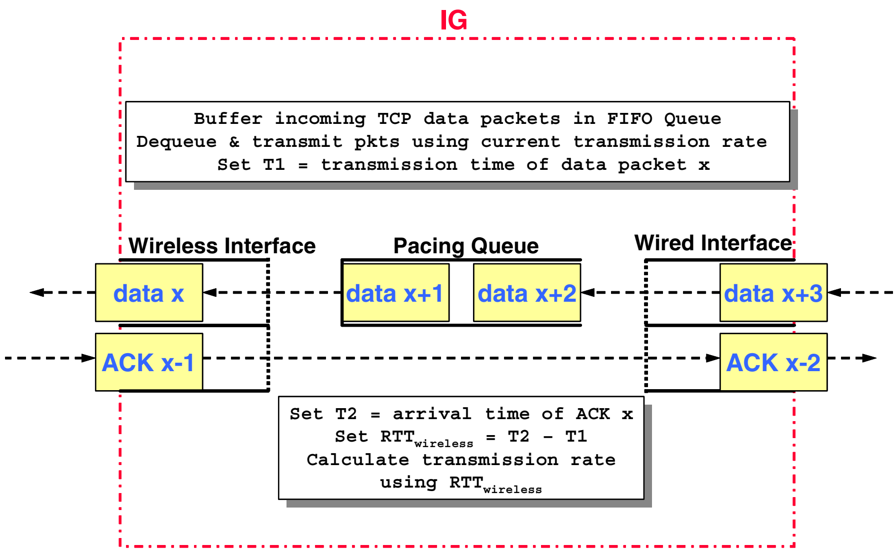

Fig. 4. The adaptive pacing procedure at the Internet gateway.

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. Hence, we only need a constant number of free buffer space which equals awnd_i for a flow i. Accordingly, the total buffer space which should be provided at the Internet gateway is given by $\sum_{i=1}^{n}\text{awnd}_i$ 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 makes up about 37 Mbytes for 200 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 a 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 frequently 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 [8].

4.4. The goodput control scheme for oncoming flows

As we will show in Section 5, applying adaptive pacing at 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 Fig. 5. 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 250 m whereas both the interference range as well as the carrier sensing range are 550 m. The distance between both chains is 400 m. 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 (FTP 1) running from the wired node B as FTP source to the wireless node A1 as FTP destination and the second (FTP 2) running from the wireless node A2 as FTP source to the wired node B as FTP destination.

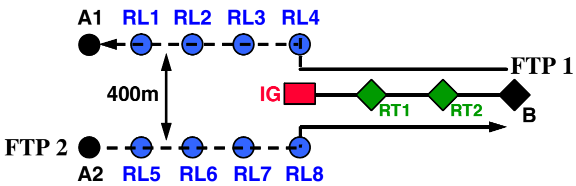

Fig. 5. The two parallel chains topology.

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, FTP 1 still achieves more goodput than FTP 2. 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 → IG mainly depends on the interaction between two effects, namely the

Table 2
Link-layer packet drops for each wireless link in 1000 s simulation time

TCP ACKs	A1 → RL1 8	RL1 → RL2 146	RL2 → RL3 112	RL3 → RL4 12	RL4 → IG 40
TCP data	A2 → RL5 11	RL5 → RL6 128	RL6 → RL7 147	RL7 → RL8 141	RL8 → IG 102
TCP data	RL1 → A1 1	RL2 → RL1 54	RL3 → RL2 90	RL4 → RL3 8	IG → RL4 40
TCP ACKs	RL5 → A2 7	RL6 → RL5 118	RL7 → RL6 127	RL8 → RL7 50	IG → RL8 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 RL7 and IG lie in each other's carrier sensing range. That means that RL7 keeps deferring its transmission while IG is transmitting to RL4. Such deferring causes a TCP timeout at the source A2 as the TCP packet is considered lost, which degrades TCP goodput. 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 a longer delay for delivering the TCP ACK packets 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 for 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 s 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 the 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 at 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. Considering handovers due to node mobility

In wireless mesh networks, there may be scenarios where the TCP entities in the wireless domain are 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.

4.6. The TCP-GAP algorithm

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. Fig. 6 outlines the functionality added to the TCP implementation at the wireless TCP sender, whereas Fig. 7 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

```
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
}
```

Fig. 6. 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
}
```

Fig. 7. Pseudo code for TCP-GAP implemented at the Internet gateway.

of hops to wireless nodes and the bandwidth of the wireless interface are provided.

5. Comparative performance study

5.1. Simulation environment

The simulation experiments in this paper are conducted using the network simulator ns-2 [10]. In the wireless domain, the link-layer parameters of IEEE 802.11 are configured to provide a transmission range of 250 m and a carrier sensing range as well as an interference range of 550 m, as consistent with a Lucent WaveLan DSSS radio interface. The transmission of each data packet on the link 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/g and set the size of TCP data packets to 1460 bytes. In the last experiment of Section 5.2.2 we also consider a higher wireless bandwidth of 54 Mbit/s as supported by IEEE 802.11a/g while varying the bandwidth of the full-duplex wired links. Unless otherwise stated, in the wireless domain of all considered topologies, each node is 200 m apart from each of its adjacent nodes. As ad hoc routing protocol for packet routing in the wireless domain we use AODV [15]. For simulations with FTP-like traffic, unless otherwise stated, we set the bandwidth of the full-duplex wired links to 10 Mbit/s and the link delay to 40 ms.

In all experiments, except for experiments showing transient behavior over time, 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 5000 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, i.e., the confidence intervals are obtained with a confidence level of 95%.

5.2. FTP-like data transfer

In the first set of scenarios, we consider TCP senders with continuous data transfer, i.e. FTP-like traffic where the TCP flow is backlogged, simulating large file transfers. Potential real life scenarios are for instance students downloading lecture materials from the university server or police patrol officers acquiring information on a suspect from the local database of their department.

5.2.1. Chain topology

First we consider a chain topology as depicted in Fig. 3 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. Fig. 8 shows the goodput vs. 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 link-layer 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 $h \geqslant 4$, 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. Fig. 9 shows that for $h < 4$, both TCP variants achieve similar goodput with a maximum of 3% value deviation. For $h \geqslant 4$, the adaptive rate-based transmission of TCP-GAP effectively decreases network congestion and achieves up to 42% more goodput than TCP NewReno.

5.2.2. Parallel chains topology

As a second topology, we consider two parallel chains as shown in Fig. 5 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

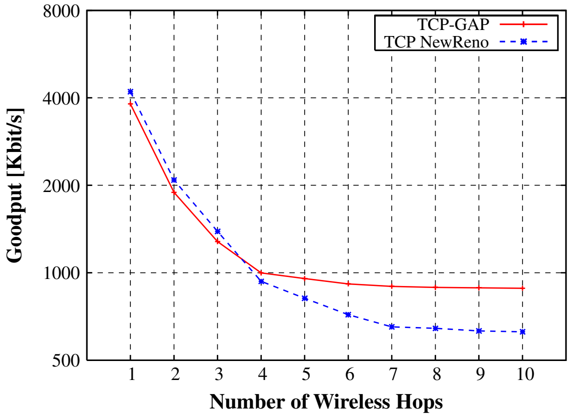

Fig. 8. TCP goodput vs. wireless chain length for wireless-to-wired flows.

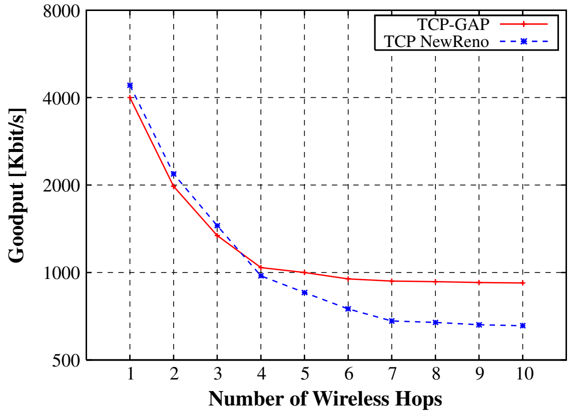

Fig. 9. TCP goodput vs. wireless chain length for wired-to-wireless flows.

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. Figs. 10–12 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 Figs. 10 and 11 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 the 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. The starvation of FTP 2 using TCP NewReno is due to the aggressive window strategy which provokes severe contention between the flows on link layer. While one flow, FTP 1 in this case, occupies almost the entire bandwidth, the other flow experiences a severe packet drop on link layer, increasing the corresponding IEEE 802.11 contention window even further. 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 400 m inter-chain distance.

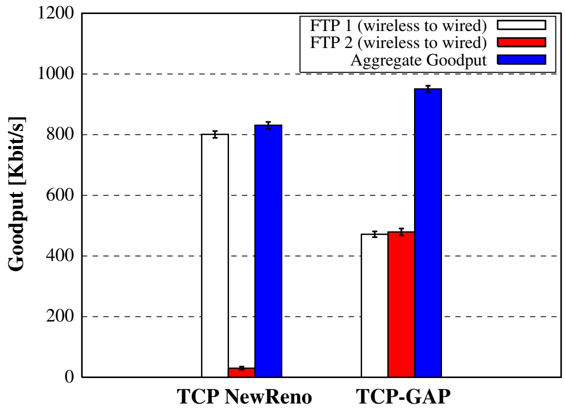

Fig. 10. Goodput in parallel chains topology for wireless-to-wired flows.

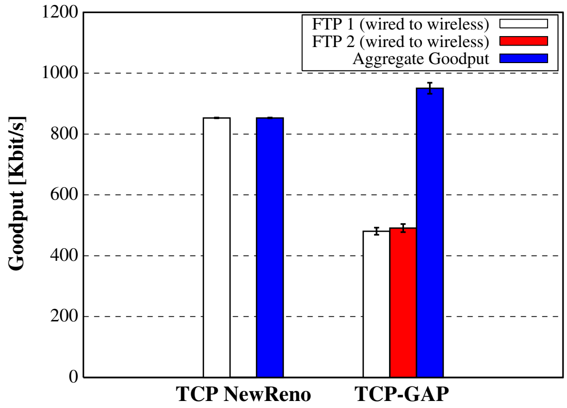

Fig. 11. Goodput in parallel chains topology for wired-to-wireless flows.

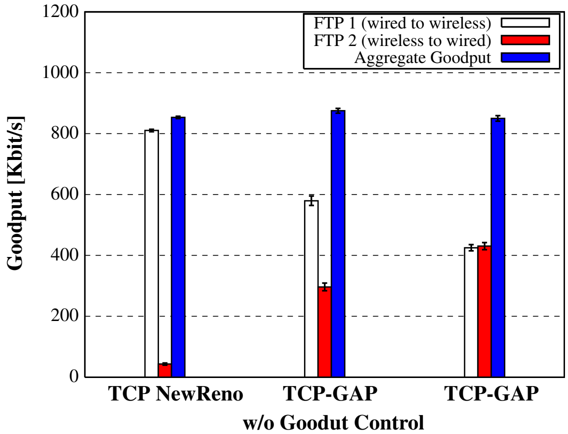

Fig. 12. Goodput in parallel chains topology for oncoming flows.

Fig. 12 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 as in the previous two cases. Such unfairness between oncoming flows was also observed in [17,19] and further discussed in Section 4. In Fig. 12 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.

5.2.2.1. 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 $N_1 = 130$ s, then restarts again at time $N_2 = 160$ s where it continues until the end. We are interested in studying how FTP 1 reacts to the stopping and starting of FTP 2. Figs. 13 and 14 plot the goodput of both flows vs. 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 contend 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. In Section 5.3 we will show how this improved responsiveness results in substantial improvement in aggregate goodput for short TCP flows.

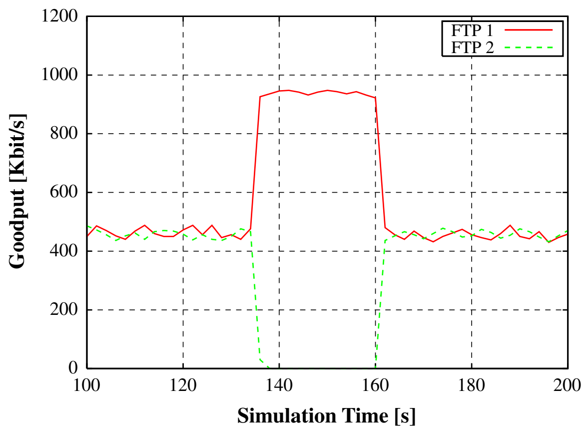

Fig. 13. Responsiveness of TCP-GAP (high responsiveness).

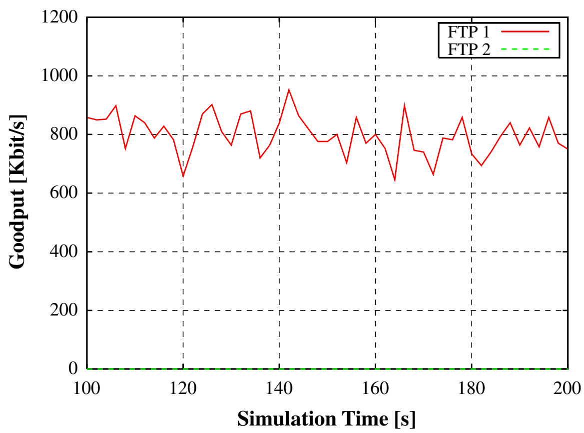

Fig. 14. Responsiveness of TCP NewReno (FTP 2 completely starves).

5.2.2.2. Sensitivity to bandwidths of the wireless/wired parts. In order to get intuition on the sensitivity of the performance of TCP to the ratio between bandwidth in the wired part to the wireless part of the connection, we consider the wired-to-wireless case as used for the responsiveness simulation, i.e., by considering two FTP flows. We adopt IEEE 802.11g in the wireless domain and set the wireless link bandwidth to 54 Mbit/s. For the wired domain, we vary the bandwidth from 1 Mbit/s up to 100 Mbit/s and plot the aggregate goodput as well as the fairness index of both TCP-GAP and TCP NewReno. As measure for fairness, we use Jain's index [13] which is given by

$$F(x) = \left[\sum_{i=1}^{n} x_i\right]^2 \Big/ n\sum_{i=1}^{n} x_i^2, \tag{6}$$

where n denotes the number of flows and x_i describes the goodput achieved by flow i.

By varying the wired link bandwidth we consider different network scenarios where we may have a simple DSL link to the Internet with 1–2 Mbit/s as well as other cases where the FTP connections run between a wireless node and a close-by campus server over fast links of 100 Mbit/s. Figs. 15 and 16 show the results of this simulation. In the figures we observe that for wired link bandwidths up to 5 Mbit/s, TCP-GAP and TCP NewReno achieve similar aggregate goodput and nearly optimal fairness index. However, for bandwidths above 5 Mbit/s, TCP-GAP achieves about 13% more aggregate goodput than TCP NewReno while maintaining optimal fairness results. Contrary to TCP-GAP, TCP NewReno fails to divide the available bandwidth equally between the flows for wired link bandwidths above 5 Mbit/s.

The reason for such behavior is that for wired link bandwidths below 5 Mbit/s, the wired part constitutes the bottleneck of the TCP connections. Thus, the wireless link is underutilized and there is no sign of congestion. This leads to similar goodput

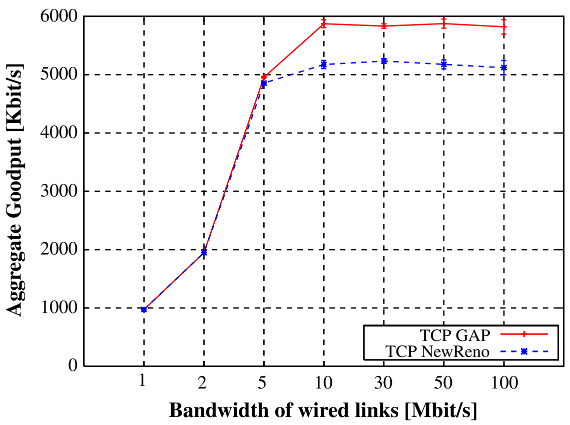

Fig. 15. Aggregate goodput vs. bandwidth of wired links.

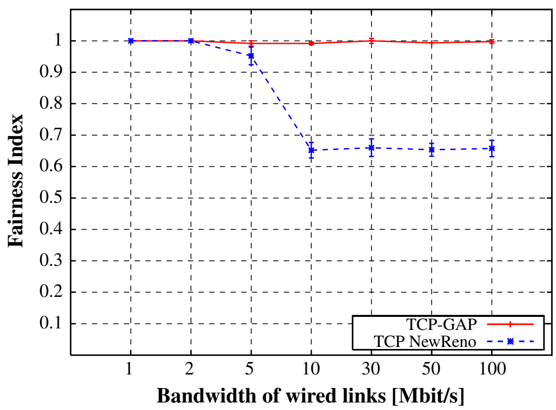

Fig. 16. Jain's fairness index vs. bandwidth of wired links (0.5 indicates worst fairness among two flows, while 1 indicates optimal fairness).

and nearly optimal fairness results. As the bandwidth of the wired link increases, the traffic load in the wireless domain also increases up to a point where the wired link bandwidth surpasses the available bandwidth in the wireless domain. From that point on the wireless domain constitutes the bottleneck of the TCP connections resulting in increased congestion. While this noticeably degrades the performance of TCP NewReno with respect to aggregate goodput and fairness, TCP-GAP achieves significantly better results due to its adaptive transmission strategy.

Note that the actually achievable bandwidth strongly depends on the number of wireless hops of the TCP connections, i.e., the achievable bandwidth decreases with increasing number of wireless hops as shown in Figs. 8 and 9. In this scenario, the achievable bandwidth is about 5 Mbit/s.

5.2.3. 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 Fig. 17. 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 Fig. 17, 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).

Figs. 18–20 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 Fig. 20, 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

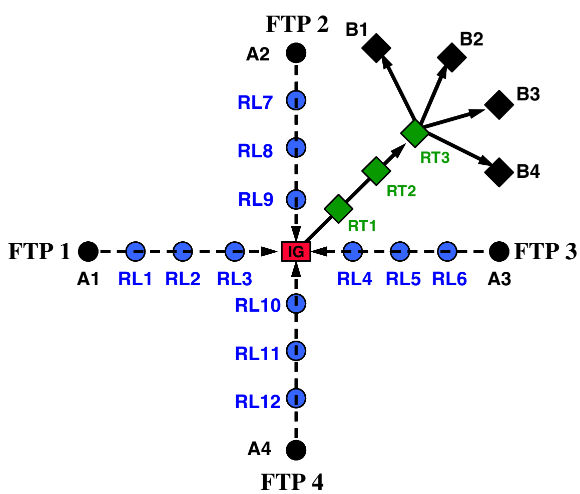

Fig. 17. The cross topology.

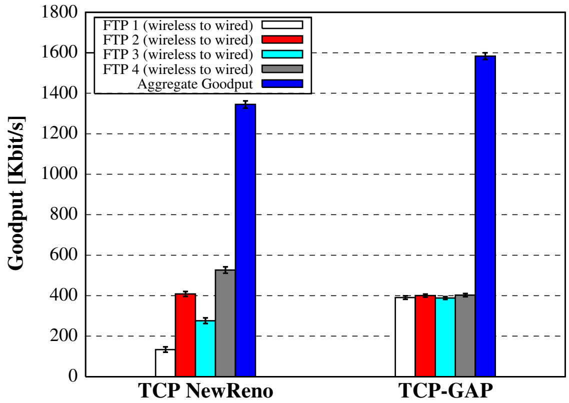

Fig. 18. Goodput in cross topology for wireless-to-wired flows.

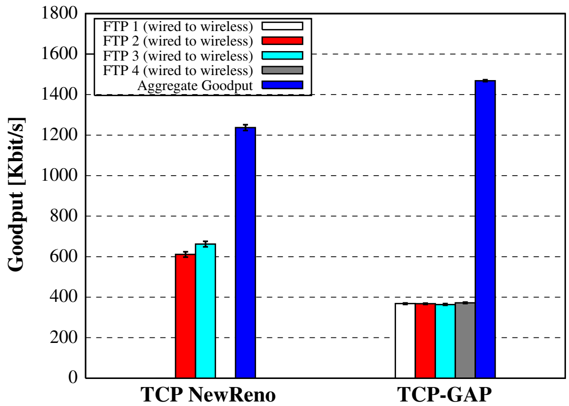

Fig. 19. Goodput in cross topology for wired-to-wireless flows.

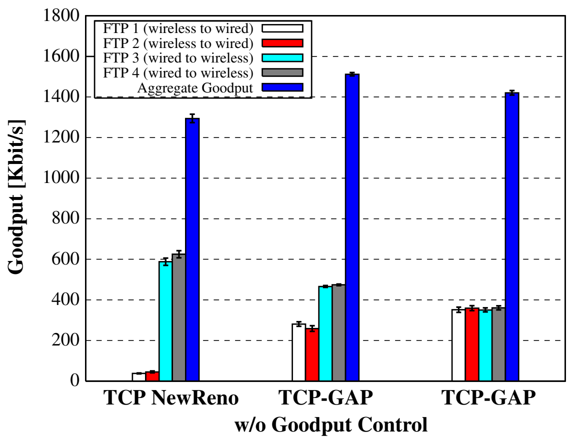

Fig. 20. Goodput in cross topology for oncoming flow.

flows, even in cases where we have multiple wireless-to-wired as well as wired-to-wireless flows.

5.2.4. Random topology

As a final topology for the continuous FTP data transfer experiments we consider a random topology of 120 wireless nodes uniformly distributed on a flat area of 2500 m × 1000 m. 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–FTP 4 run from the wired to the wireless domain and FTP 5–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 Fig. 5. Thereby, the wired host B3 constitutes the TCP source for FTP 1–FTP 4 and the TCP destination for FTP 5–FTP 8. Opposed to previous experiments, in this simulation, TCP flows run on paths of different lengths. Fig. 21 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 [18]. Note that the different wireless path lengths

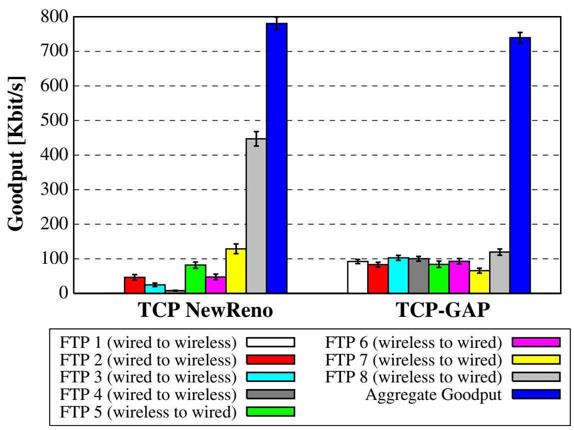

Fig. 21. Goodput in random topology for oncoming flows running on paths of different lengths.

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.

5.3. HTTP-like data transfer

In this set of experiments, we consider variable length TCP flows, i.e., HTTP-like data transfer where a TCP source transmits small files with variable pause times between successive file transfers. Following [8], we choose Pareto distributed file sizes with mean 30 kbytes and shape factor $\beta = 1.5$, whereas the pause times between successive file transfers are exponentially distributed. Specifically, file sizes and pause times are generated using an external traffic generator, which utilizes the GNU Scientific Library [20]. File sizes are generated using the *gsl_ran_pareto* function with shape factor $\beta = 1.5$ and mean 30 kbytes, whereas pause times are generated using the *gsl_ran_exponential* function with variable values for the mean. In all HTTP experiments, we only consider the case where the flows run from the wired domain to the wireless domain, which is consistent to the case in reality where most of the traffic of an HTTP connection constitutes the download of Web data from a WWW server in the Internet.

5.3.1. Parallel chains topology

We reconsider the parallel chains topology introduced in Section 5.2.2, though, with two variable length HTTP flows rather than continuous FTP flows. To evaluate the impact of inter-file pause times on TCP performance, we vary the mean for exponentially distributed pause times and plot the fairness index as well as the aggregate averaged goodput of TCP-GAP and TCP NewReno. The aggregate averaged goodput is computed by measuring the goodput of each file transfer, averaging the goodput of the file transfers for each flow and finally summing up this average goodput over all flows. The goodput of each file transfer is computed similar to the case of FTP flows, since pause times between the file transfers are not considered.

Figs. 22 and 23 show the results of this simulation. In Fig. 22 we see that TCP-GAP outperforms TCP NewReno, achieving up to 70% more aggregate averaged goodput. In particular, we observe that the aggregate averaged goodput of TCP-GAP actually increases for increasing pause times, while the goodput of TCP NewReno remains unchanged.

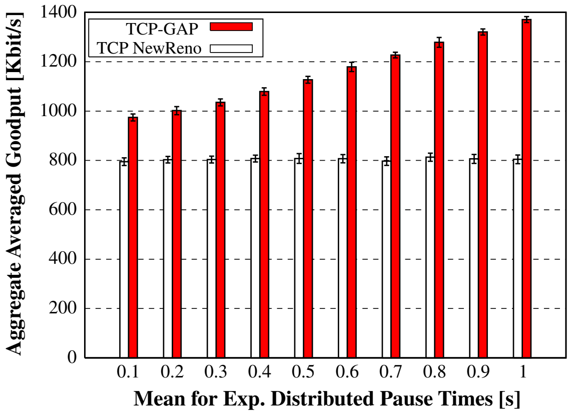

Fig. 22. Aggregate averaged goodput vs. exp. distributed pause times.

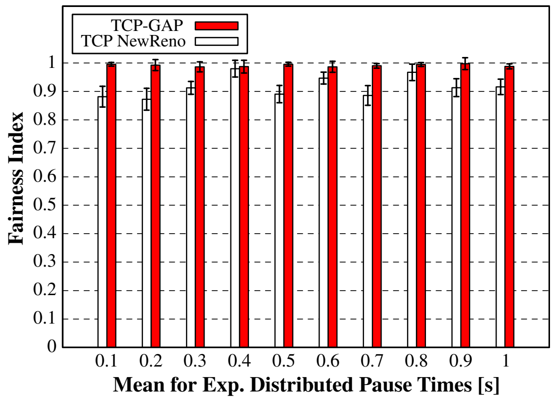

Fig. 23. Jain's fairness index vs. exponentially distributed pause times (0.5 indicates worst fairness among two flows, while 1 indicates optimal fairness).

Similar effects were also identified in [8] and are to be ascribed to the high responsiveness of the adaptive pacing scheme. That is, considering HTTP traffic, the probability that both flows contend for the wireless channel at the same time decreases with increasing pause times. Thus, a single flow gets a higher chance to acquire the entire available bandwidth for itself without sharing it with the other flow. Hence, since TCP-GAP is highly responsive to changing network conditions, it takes advantage of this effect and can quickly acquire the available bandwidth. On the contrary, the bad responsiveness of TCP NewReno does not allow taking advantage of the free bandwidth and, thus, the aggregate averaged goodput does not change with increasing pause times.

Fig. 23 underlines the fact that the contention between flows decreases for HTTP-like traffic, since

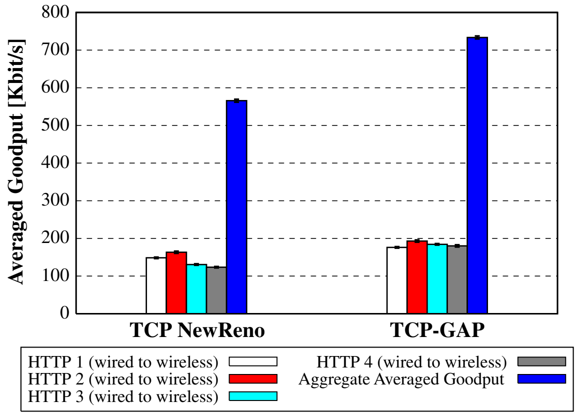

Fig. 24. Averaged goodput for HTTP random scenario.

the fairness of TCP NewReno increases considerably compared to the case for FTP-like data transfer where TCP NewReno has the worst fairness index of 0.5. However, the fairness index of TCP-GAP still lies above the fairness index of TCP NewReno for all pause times. From the previous figures we conclude that TCP-GAP achieves nearly optimal fairness, not only for FTP-like traffic, but also for HTTP-like traffic.

5.3.2. Random topology

As a final scenario we consider another random topology with the same settings as given in Section 5.2.2.2, however, in this case we consider four HTTP-like flows with a fixed mean for the exponentially distributed pause times of 1 s. Consistent with our previous findings, in Fig. 24 we observe that TCP NewReno achieves good fairness between the HTTP flows due to the decreased channel contention which results from the relatively high inter-file pause times. However, TCP-GAP still achieves slightly better fairness and considerably more aggregate averaged goodput than TCP NewReno. From the simulations with HTTP-like traffic we conclude that Gateway Adaptive Pacing significantly improves TCP performance for HTTP traffic both in terms of fairness and goodput.

6. Conclusion

For improving both goodput and fairness of TCP flows in multihop wireless networks with Internet connectivity, we proposed an adaptive pacing scheme on the Internet gateway and the wireless TCP sender. By deploying rate-based congestion control for the wireless part of the network at the Internet gateway, our approach, denoted as TCP with Gateway Adaptive Pacing (TCP-GAP), accounts for the different characteristics of the wireless and wired domains. 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 through comprehensive simulations using ns-2 [10] that nearly optimal fairness between multiple TCP flows in hybrid wireless/wired networks can be achieved by solely modifying the transport layer. Thus, our approach is easily deployable since it requires neither modifications of standard TCP in the wired domain nor modifications at the link or network layers. Furthermore, TCP-GAP is fully TCP compatible and does not impose any control traffic overhead. In future work, we are developing a prototype implementation of TCP-GAP in our wireless mesh testbed.

References

[1] A. Aggrawal, S. Savage, T. Anderson, Understanding the performance of TCP pacing, in: Proceedings of the IEEE INFOCOM, Tel Aviv, Israel, 2000.
[2] Ö.B. Akan, 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, W. Wang, Wireless mesh networks: a survey, Computer Networks 47 (2005).
[4] H. Balakrishnan, V. Padmanabhan, S. Seshan, 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, in: Proceedings of the ACM MOBIHOC, Lausanne, Switzerland, 2002.
[6] J. Bicket, D. Aguayo, S. Biswas, R. Morris, Architecture and evaluation of an unplanned 802.11b mesh network, in: Proceedings of the ACM MOBICOM, Cologne, Germany, 2005.
[7] D. De Couto, D. Aguayo, J. Bicket, R. Morris, A high-throughput path metric for multi-hop wireless routing, in: Proceedings of the ACM MOBICOM, San Diego, CA, 2003.
[8] S. ElRakabawy, A. Klemm, C. Lindemann, TCP with adaptive pacing for multihop wireless networks, in: Proceedings of the ACM MOBIHOC, Urbana-Champaign, IL, 2005.
[9] S. ElRakabawy, A. Klemm, C. Lindemann, Gateway adaptive pacing for TCP across multihop wireless networks and the internet, in: ProcACM/IEEE MSWiM, Malaga, Spain, 2006.

[10] K. Fall, 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, M. Gerla, The impact of multihop wireless channel on TCP throughput and loss, in: Proceedings of the IEEE INFOCOM, San Francisco CA, 2003.
[12] V. Gambiroza, B. Sadeghi, E. Knightly, End-to-end performance and fairness in multihop wireless backhaul networks, in: Proceedings of the ACM MOBICOM, Philadelphia, PA, 2004.
[13] R. Jain, D. Chiu, W. Hawe, A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Systems, DEC Technical Report DEC-TR-301, 1984.
[14] S. Mascolo, C. Casetti, M. Gerla, M. Sandidi, R. Wang, TCP Westwood: bandwidth estimation for enhanced transport over wireless links, in: Proceedings of the ACM MOBICOM, Rome, Italy, 2001.
[15] C. Perkins, E. Royer, S. Das, Ad hoc on-demand distance vector (AODV) routing, in: IETF RFC 3561, 2003.
[16] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, R. Sivakumar, ATP: a reliable transport protocol for ad hoc networks, in: Proceedings of the ACM MOBIHOC, Annapolis, MA, 2003.
[17] K. Xu, S. Bae, S. Lee, M. Gerla, TCP behavior across multihop wireless networks and the wired internet, in: Proceedings of the ACM WoWMoM, Atlanta, GA, 2002.
[18] K. Xu, M. Gerla, L. Qi, Y. Shu, Enhancing TCP fairness in ad hoc wireless networks using neighborhood RED, in: Proceedings of the ACM MOBICOM, San Diego CA, 2003.
[19] L. Yang, W. Seah, Q. Yin, Improving fairness among TCP flows crossing wireless ad hoc and wired networks, in: Proceedings of the ACM MOBIHOC, Annapolis MD, 2003.
[20] The GNU Scientific Library. <http://www.gnu.org/software/gsl/>.

Sherif M. ElRakabawy received the degree Diplom-Informatiker (M.Sc. in Computer Science) from the University of Bonn, Germany in 2003. Currently he is a Ph.D. student at the Chair of Computer Networks and Distributed Systems at the University of Leipzig. His research interests include mobile ad hoc/mesh networks as well as mobile peer-to-peer systems.

Alexander Klemm received the degree Diplom-Informatiker (M.Sc. in Computer Science) from the University of Dortmund in March 2000. From April 2004 to June 2006 he was a Ph.D. student in the Mobile Computing Systems Group at the University of Dortmund, obtaining his Ph.D. in Computer Science in June 2006. Currently he is a project manager at radprax MVZ GmbH.

Christoph Lindemann holds the Chair of Computer Networks and Distributed Systems in the Department of Computer Science at the University of Leipzig. He received the degree Diplom-Informatiker (M.S. in Computer Science) from the University of Karlsruhe, Germany in 1988 and the degree Doktor-Ingenieur (Ph.D. in Engineering) from the Technische Universität Berlin, Germany in 1992. His current research interests lie in mobile computing systems, especially mobile ad hoc networks and peer-to-peer systems as well as modeling and performance evaluation as an umbrella topic. He is member of the IFIP working group 7.3 and a senior member of the IEEE. He is on the editorial board of the international journals Ad Hoc Networks and Performance Evaluation. He is currently serving as chair of the special interest group on measurements, modeling, and evaluation of computer systems and communication networks within the German Society of Informatics (GI). In 2005, he served as general co-chair for the 11th International Conference on Mobile Computing and Networking, ACM MobiCom. He organized the ACM MobiShare Workshop in 2006 and is serving as general chair of the 26th International Symposium on Computer Performance, Modeling, Measurements, and Evaluation, Performance 2007.