TCP with Adaptive Pacing for Multihop Wireless Networks*

Sherif M. ElRakabawy, Alexander Klemm, and Christoph Lindemann
University of Dortmund
Department of Computer Science
August-Schmidt-Str. 12
44227 Dortmund Germany
http://mobicom.cs.uni-dortmund.de/

ABSTRACT
In this paper, we introduce a novel congestion control algorithm for TCP over multihop IEEE 802.11 wireless networks implementing rate-based scheduling of transmissions within the TCP congestion window. We show how a TCP sender can adapt its transmission rate close to the optimum using an estimate of the current 4-hop propagation delay and the coefficient of variation of recently measured round-trip times. The novel TCP variant is denoted as TCP with Adaptive Pacing (TCP-AP). Opposed to previous proposals for improving TCP over multihop IEEE 802.11 networks, TCP-AP retains the end-to-end semantics of TCP and does neither rely on modifications on the routing or the link layer nor requires cross-layer information from intermediate nodes along the path. A comprehensive simulation study using ns-2 shows that TCP-AP achieves up to 84% more goodput than TCP NewReno, provides excellent fairness in almost all scenarios, and is highly responsive to changing traffic conditions.

Categories and Subject Descriptors
C.2.0 [**Computer Communication Networks**]: General – *data communications.*

General Terms
Algorithms, Design, Performance.

Keywords
Analysis and design of transport protocols, IEEE 802.11 wireless networks, End-to-end congestion control, Performance evaluation.

1. INTRODUCTION
Multihop wireless networks using IEEE 802.11 possess several properties, which are different to the wired Internet for which the widely deployed TCP NewReno implementation has been optimized. Opposed to wired networks, in IEEE 802.11 networks the wireless channel is a scarce resource shared among nodes within their radio range. Furthermore, channel capture, hidden and exposed terminal effects, and the IEEE 802.11 medium access control constitute features of wireless multihop networks not present in a wired IP network. In fact, for multihop wireless networks, most losses experienced by TCP are due to packet drops at the link layer and not due to buffer overflow [9]. Furthermore, since the congestion control of TCP NewReno is based on lost data packets, the size of its congestion window is overshooting rather than proactively sense incipient congestion by monitoring the network traffic. Because of all these features, TCP NewReno possesses quite poor performance in multihop wireless networks, as well as exhibits severe unfairness among competing TCP flows.

Improving the performance of reliable data transport in mobile ad hoc networks (MANET) by explicit rate control has been explored in several non-TCP protocols e.g., [5], [17]. Instead of sending new packets into the network only when old packets have been acknowledged, these approaches send packets at a pre-determined rate. TCP Pacing is a hybrid between a pure rate-based transmission control and TCP's use of the congestion window to trigger new data packets to be sent into the network. Based on the intuition that bursty traffic produces higher queuing delays, more packet losses and lower goodput, smoothing the behavior of TCP traffic by pacing shall result in improved performance and fairness.

Aggarwal, Savage, and Anderson presented a comprehensive evaluation of TCP with pacing for the Internet [2]. They considered an implementation of pacing based on a leaky bucket algorithm with evenly spaced timeouts. As duration of the timeouts they considered the ratio of the averaged measured round trip time and the current window size. Their study showed that contrary to intuition for the Internet, TCP Pacing often exhibits significantly less goodput than regular TCP. In fact, TCP Pacing exhibits worse performance than regular TCP for scenarios with large buffers due to delaying congestion signals of the network. On the other hand, pacing improves TCP fairness and packet drop rates when round trip times are highly variable as typical for multihop wireless networks.

In this paper, we introduce a novel congestion control algorithm for TCP over multihop IEEE 802.11 networks implementing rate-based scheduling of transmissions within the TCP congestion window. Rather than evenly spacing the transmissions of packets within the congestion window as proposed in [2], in our approach, a TCP sender adaptively sets its transmission rate using an estimate of the current 4-hop propagation delay and the coefficient of variation of recently measured round-trip times. The *4-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 4 hops apart from the source node along the path to the destination. The novel TCP variant is denoted as *TCP with Adaptive Pacing (TCP-AP)*. Opposed to previous proposals for improving TCP over multihop IEEE

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

MobiHoc'05, May 25–27, 2005, Urbana-Champaign, Illinois, USA.

802.11 networks, TCP-AP retains the end-to-end semantics of TCP and does neither rely on modifications on the routing or the link layer nor requires cross-layer information from intermediate nodes along the path. A comprehensive simulation study using ns-2 [8] shows that TCP-AP achieves up to 84% more goodput than TCP NewReno, provides excellent fairness in almost all scenarios, and is highly responsive to changing network traffic conditions. In particular, TCP-AP provides fair sharing of the available bandwidth, even when competing flows are not within each other's transmission range, but within each other's interference range, since the proposed algorithm relies on the end-to-end measurement of the interference experienced by a TCP connection.

The remainder of this paper is organized as follows. Section 2 summarizes related work on TCP for multihop wireless networks and Section 3 describes the settings of the simulation environment used throughout this paper. In Section 4, we introduce the novel TCP congestion control algorithm. A comprehensive performance study of TCP-AP versus TCP NewReno with and without dynamically delaying acknowledgements is presented in Section 5. Finally, concluding remarks are given.

2. RELATED WORK

Fu et al. [9] pointed out the hidden terminal problem in wireless multihop networks and experimentally showed that for a chain topology the optimal windows size, for which TCP achieves best throughput, is roughly given by 1/4 of the hop count of the path. Furthermore, they proposed two enhancements on the link layer: adaptive pacing to distribute traffic on the link layer among intermediate nodes in a more balanced way and link layer RED to throttle TCP senders when incipient congestion is detected. Using simulation, they showed that depending on the scenario, these link layer enhancements improve TCP goodput by 5% to 30% due to better spatial reuse. Xu et al. [18] proposed the neighborhood RED (NRED) scheme on routing layer to throttle TCP senders when incipient congestion is detected, by purposely dropping TCP packets on intermediate nodes. Nodes forming a neighborhood manage a virtual distributed queue in order to coordinate the packet drops of individual nodes. Using simulation, the authors showed that NRED could substantially improve fairness in multihop wireless networks.

Our approach differs fundamentally from [9], [18]. TCP-AP just requires slight modifications on the transport layer and does neither require modifications on the routing layer as [18] and link layer as [9], nor extra communication between neighboring nodes. As a consequence, TCP-AP can be incrementally deployed. Furthermore, TCP-AP integrally improves both fairness and goodput without provoking packet losses.

Sundaresam et al. [17] and Chen et al. [5] introduced two new special-purpose transport protocols for multihop wireless networks. Both protocols employ pure rate-based transmission of packets, where the transmission rate is determined using feedback from intermediate nodes along the path. In [17], the authors propose to dynamically adjust the transmission rate according to the maximum packet queuing delay on intermediate nodes along the network path. Chen et al. [5] also proposed an explicit rate-based flow control scheme for multihop wireless network. Using cross-layer information from both the MAC and the routing layer, the sending rate of a flow is conveyed from intermediate nodes along the path in special control headers attached to each data packet.

In contrast to [5], [17], TCP-AP retains the end-to-end semantics of TCP without relying on any cross-layer information from intermediate nodes along the path. As a consequence, TCP-AP can be incrementally deployed, since TCP-AP is not only TCP-friendly, but also TCP compatible.

Altman and Jiménez [1] proposed a dynamic scheme for delaying ACKs in order to improve TCP throughput in multihop wireless networks. Using simulation, they showed that for an *h*-hop chain, delaying ACKs yields around 50% more throughput for TCP NewReno. Building upon their results, we are also considering dynamically delaying ACK, though, not only for TCP NewReno, but also for TCP-AP. Furthermore, we evaluate the dynamic delayed ACK scheme for a comprehensive set of network topologies rather than just for an *h*-hop chain.

Several authors introduced TCP enhancements for coping with mobility in ad hoc wireless networks over IEEE 802.11. Holland and Vaidya [11] introduced explicit link failure notification (ELFN) as a feedback mechanism from the network in order to help TCP distinguish congestion losses and losses induced by link failures. Yu [20] proposed two cross-layer communication mechanisms that further improve TCP performance in case of packet losses due to mobility. We focus on TCP performance and fairness in static wireless networks instead, though, our results may well be utilized together with their findings in order to adapt TCP-AP for mobile wireless multihop networks.

3. SIMULATION ENVIRONMENT

We conduct simulation experiments using the network simulator ns-2 [8]. All 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. Thus, our setting is consistent with real wireless networks, in which the transmission range of a node is typically smaller than its interference range. 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 channel bandwidth of 2 Mbit/s and set the size of TCP and UDP data packets to 1460 bytes. For all nodes, we assume a buffer size of 50 packets. Unless otherwise stated, in all considered topologies, each node is 200 meters apart of each of its adjacent nodes. As ad hoc routing protocol we choose AODV [16].

In order to isolate the IEEE 802.11 MAC-induced deficiencies of TCP in multihop wireless networks we only consider static scenarios throughout this paper. Note that route failures do not only occur in mobile, but also in static scenarios due to the interaction between the MAC and the routing layers in case of congestion [19]. Thus, we conjecture that in scenarios with low mobility (e.g. pedestrian speed), the performance of TCP-AP is similar to the performance in static scenarios.

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 either TCP or UDP connections, dependent on the scenario, 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.

4. RATE BASED CONGESTION CONTROL FOR TCP

4.1 Motivation

Several researchers identified the interaction of TCP with the underlying routing and MAC layers as key factor for the poor performance of TCP in IEEE 802.11 multihop wireless networks. If we neglect mobility-related problems of TCP in such networks, most important deficiencies of TCP arise from TCP's congestion control algorithm. First, TCP's window-based congestion control leads to packet bursts when received acknowledgments trigger the transmission of several data packets, e.g., when receiving a cumulative ACK. Due to the spatial reuse constraint of the wireless channel in IEEE 802.11 multihop wireless networks, neighboring nodes cannot transmit simultaneously. Thus, packet bursts result in increased contention on the wireless channel. This link layer contention may lead to packet drops due to the hidden and exposed terminal problems [18]. Second, TCP's congestion control algorithm relies on packet losses as indication of congestion and, thus, provokes losses in order to identify spare bandwidth. In IEEE 802.11 multihop wireless networks, this behavior results in increased congestion, causing significant performance degradation for TCP [9]. Recall that network congestion often triggers (false) route failures, even in static wireless networks, since the routing protocol cannot distinguish between a packet loss due to congestion and a packet loss due to a broken route [19].

To overcome both deficiencies stated above while preserving TCP compatibility, we propose to incorporate a rate-based transmission algorithm into TCP's window-based congestion control. Our approach is denoted as *TCP with Adaptive Pacing (TCP-AP)*. The problem of packet bursts is solved by spreading the transmission of successive data packets according to the computed transmission rate, which accounts for the spatial reuse constraint in IEEE 802.11 multihop wireless networks. Furthermore, by proactively identifying incipient congestion, i.e. before congestion-related losses actually occur, TCP-AP is able to adjust the transmission rate and, hence, reduce contention on the MAC layer. In contrast to TCP Pacing for the Internet [2], where the transmission of a window of packets is evenly spread over the duration of a round trip time (RTT), our approach schedules the transmission of packets based on both the size of the congestion window and the computed transmission rate. As long as the size of the congestion window is larger than the number of packets in flight, new packets are scheduled for transmission according to the current transmission rate.

4.2 Identification of Incipient Congestion

Due to its end-to-end semantics, TCP's congestion control algorithm is based on the measurement of round trip times and packet losses. In fact, in current TCP variants such as Reno and NewReno, the actual identification of congestion is solely laid upon the observation of packet losses. Therefore, TCP increases the load issued into the network until a packet loss is detected, where such packet loss identifies congestion. Note that although TCP Vegas uses RTT measurements in addition to packet losses to identify congestion, it still suffers from bursty packet transmissions in wireless multihop networks.

Considering the characteristics of IEEE 802.11 multihop networks, it becomes obvious that a transport protocol which actually provokes packet drops to get network feedback has to suffer from poor performance. Thus, our goal is to develop a congestion control algorithm that identifies high contention on the network path of the TCP connection and proactively throttles the transmission rate before losses occur. In order to retain the end-to-end semantics of TCP, such congestion control algorithm requires a measure obtainable at the TCP entities, which quantifies the degree of contention on the network path. We propose the coefficient of variation of recently measured round trip times, cov_{RTT}, as key measure for the degree of the contention on the network path. This measure is given by:

$$cov_{RTT} = \frac{\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(RTT_i - \overline{RTT})^2}}{\overline{RTT}}$$

Here, N is the number of considered RTT samples, $\overline{RTT}$ is the mean of the samples, and RTT_i denotes the value of the *i-th* RTT sample.

To justify the applicability of the coefficient of variation cov_{RTT} as an appropriate measure for identifying contention, we analyze the influence of network contention on the development of the RTT samples. Using simulation, we observe that the fluctuation of the RTT samples, which can be quantified by cov_{RTT}, provides a reliable measure for the degree of contention. As first basic topology, we consider a cross scenario of two chains, each of five nodes, as illustrated in Figure 1. On the horizontal chain, a TCP connection with a fixed transmission rate of 100 KBit/s is used to measure the RTT samples at the TCP sender. On the vertical chain, a UDP connection with varying transmission rates produces background traffic. The background traffic starts at time T_1=30s and stops at time T_2=60s. Figures 3 to 5 show the influence of the background traffic on the RTT samples of the TCP connection. The left graph of each figure plots the measured RTT samples of the TCP connection for background traffic of 150 KBit/s, 200 KBit/s, and 250 KBit/s, respectively. The right graph shows the corresponding coefficient of variation cov_{RTT}. Note that the available bandwidth in this topology lies above 300 KBit/s. Although the offered load of both traffic sources is far below the available bandwidth in Figure 3, the TCP connection experiences significant fluctuation in the RTT samples when its flow competes for the shared channel with the UDP background flow. With increasing background traffic, we observe increasing fluctuation in the measured RTT samples, as shown in Figures 4 and 5.

As a second basic topology, we consider two parallel chains with a distance of 400m as shown in Figure 2. Consistent with the cross topology, we regard a TCP flow with fixed rate on one chain and a variable-rate UDP flow on the other chain. Note that since both

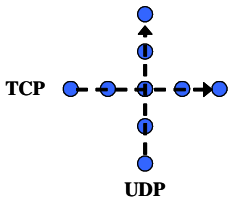

Figure 1: Cross topology

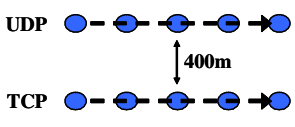

Figure 2: Symmetric parallel chains topology

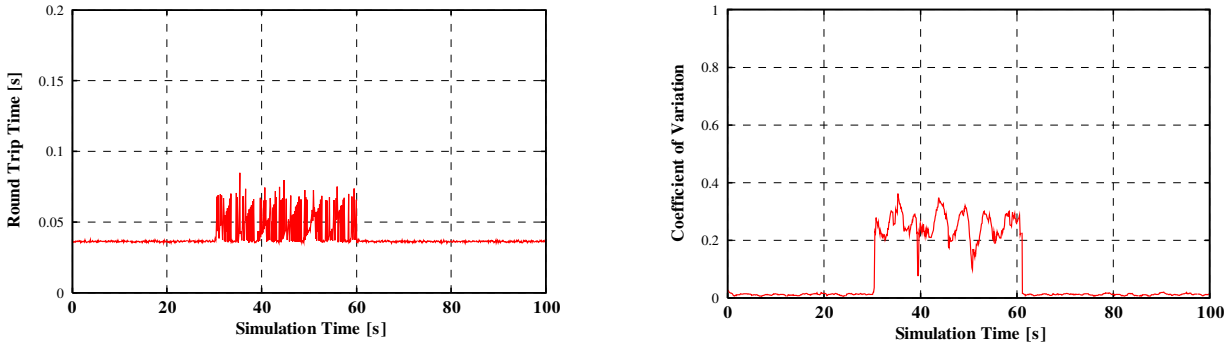

Figure 3: Influence of UDP background traffic with 150 KBit/s on RTT samples

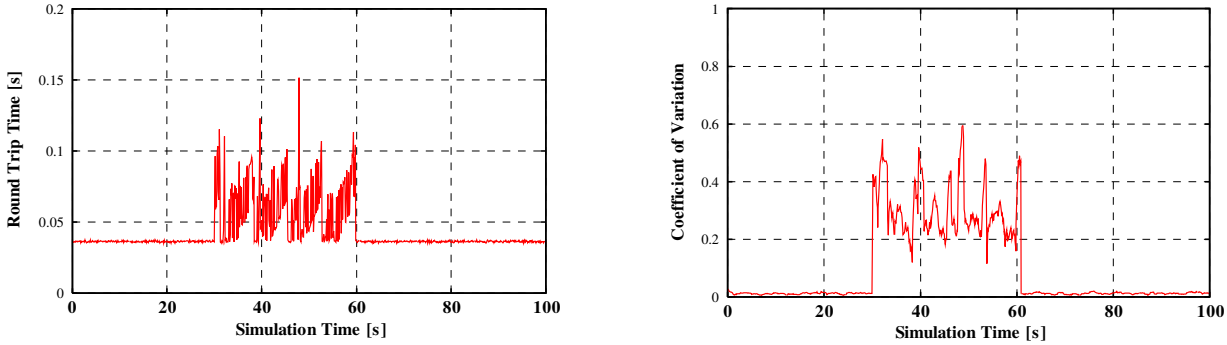

Figure 4: Influence of UDP background traffic with 200 KBit/s on RTT samples

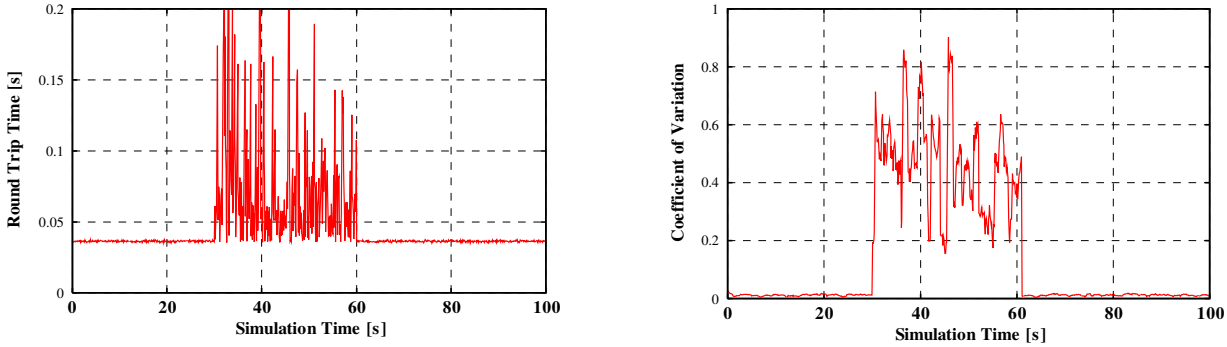

Figure 5: Influence of UDP background traffic with 250 KBit/s on RTT samples

chains lie within each other's interference range, they compete for the same wireless medium. However, no scheduling can be achieved on the MAC layer, since both chains are placed beyond each other's transmission range. That is, nodes belonging to one chain cannot hear the RTS-CTS sequence transmitted by nodes belonging to the other chain. Simulation results obtained for this topology are consistent with the results of the cross scenario. Due to space limitations, we omit the corresponding curves. Since a larger number of scenarios comprise of the cross and the parallel chain topology as basic building blocks, we conjecture that the coefficient of variation cov_{RTT} is an appropriate measure for identifying contention in multihop wireless networks. This conjecture is experimentally verified in Section 5 for a wide range of scenarios. Recall that the measure cov_{RTT} can be obtained without provoking congestion or packet losses.

4.3 The Spatial Reuse Constraint

Besides the measure of contention on the network path, the derivation of an appropriate transmission rate should also account for the spatial reuse constraint of IEEE 802.11 multihop wireless networks. That is, due to the hidden terminal effect, in a chain topology where the transmission range of each node is about 250m and the interference and carrier sensing ranges are 550m, a TCP sender at node i can only transmit a packet successfully as soon as node (i+3) has finished its transmission. We refer to the time elapsed between transmitting a TCP packet by node i and receiving the packet at node (i+4) as the *4-hop propagation delay (FHD)*.

We recall the spatial reuse constraint of IEEE 802.11 multihop networks, which is reported in previous studies such as [9], by considering the static chain topology depicted in Figure 6 where the inter-node distance is 200m. Assume that node 1 wishes to

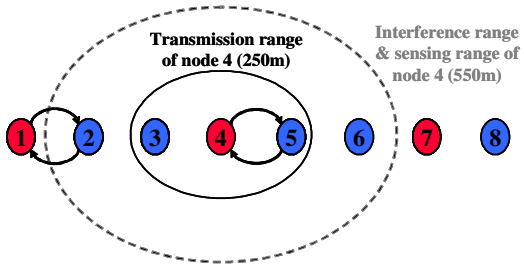

Figure 6: A chain of nodes showing the hidden terminal effect

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.

Note that such hidden terminal effects depend mainly on the characteristics of the network interface as well as the adopted routing protocol. First, the network interface determines the ratio between the transmission range and the interference range. Due to the settings of the network interface considered in this paper, hidden terminals along the path can be avoided if a transmitting node delays the transmission of a data packet until the previously sent packet is forwarded 4 times. Thus, we consider *FHD* for the calculation of the transmission rate.

Second, the adopted routing protocol determines the length of the route in terms of hop count, depending on the employed routing metrics. In this paper, we consider routing protocols which aim to minimize the hop count of the route such as AODV [16]. For this class of routing protocols the node density does not affect the choice of *FHD* as the key factor of the spatial reuse constraint. To illustrate this independence, we consider the simple chain topology depicted in Figure 7(a). The figure shows the process of route determination between source node 1 and destination node 8 in a chain topology with different inter-node distances. A potential route, which may be built by AODV is given by the nodes 1-3-6-8. Since the remaining nodes are not part of the route, they do not contribute to the spatial reuse constraint factor. Thus, the node density in a given network is irrelevant for *FHD* after the route

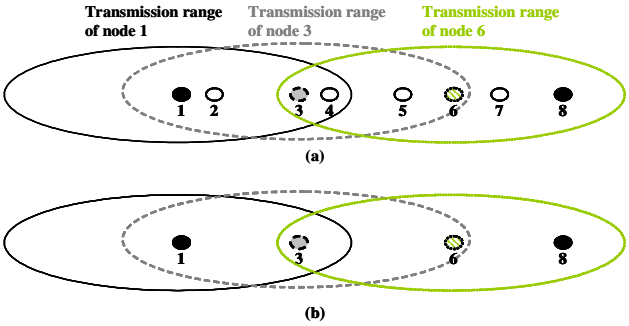

Figure 7: Example of a route determination with different inter-node distances: actual node topology (a) and logical node topology (b)

Table 1: Parameters for the adaptive computation of the transmission rate

Parameter	Meaning
h	Number of hops from sender to receiver
b	Bandwidth of the wireless interface
t_q	Average packet queuing delay per node
s_{data}	Size of TCP data packet
s_{ACK}	Size of TCP ACK packet
t_{data}	Transmission time for TCP data packet
t_{ACK}	Transmission time for TCP ACK packet
RTT	Current round trip time of TCP packets
cov_{RTT}	Coefficient of variation of RTT samples
FHD	Current sample of 4-hop propagation delay
$\widehat{FHD}$	EWMA of 4-hop propagation delay

has been established. The logical topology, which is actually used by the routing protocol, is depicted in Figure 7(b).

Note that, since we derive *FHD* from the measured round trip times of the TCP packets (RTT), our approach may be arbitrarily adjusted to estimate propagation delays for other number of hops (i.e. *n*-hop delay), e.g. in order to adapt to different settings of the network interface as well as different routing protocols.

If we assume a network with perfect scheduling on the MAC layer, the maximum spatial reuse with minimum collisions can be achieved with a transmission rate $R_{max}=1/FHD$. In fact, this transmission rate reflects the upper bound of the bandwidth-delay product for IEEE 802.11 multihop wireless networks. Following [6], an upper bound for the capacity of a path with h hops in an IEEE 802.11 multihop wireless network is given by $h/4$ packets. Let $T_{one\text{-}way}$ denote the time a packet travels from the sender to the receiver. This quantity can be computed as $T_{one-way} = FHD \cdot h/4$. Subsequently, the number of packets in flight on the way from the TCP sender to the TCP receiver with a sender's transmission rate of R_{max}, is given by:

$$\#\text{ packets in flight} = R_{\max} \cdot T_{one-way} = \frac{1}{4}h$$

Thus, the number of packets in flight transmitted with the maximum transmission rate R_{max} reflects the maximum capacity of the network path. Note that for network paths with $h < 4$, R_{max} is computed using the h-hop propagation delay instead of the 4-hop propagation delay. Without loss of generality and for ease of exposition, we only consider network paths with $h \geq 4$ in the subsequent discussion.

In order to use R_{max} as an upper bound for the transmission rate, we need to measure the 4-hop propagation delay *FHD* of the TCP data packets. Since the end-to-end semantics of TCP does not allow using information available on intermediate nodes, we estimate *FHD* based on RTT measurements at the TCP sender. Our estimation algorithm is based on two assumptions: (1) Contention on the path from the sender to the receiver is similar to the contention on the backward path and (2) transmission delays of packets (not including queuing delays) are proportional to the size of the packets. The first assumption is reasonable, since in most MANET routing protocols network paths are bidirectional, i.e., the forward and the backward paths are the same. Even when

the forward and backward paths are different, it is most likely that both paths lie within each other's interference range and do not allow concurrent transmissions. Assumption (2) results from the medium access method used in IEEE 802.11, which allows the exclusive transmission at a fixed bandwidth (e.g., 2 Mbit/s) once the sender has acquired the channel.

The RTT is composed of the sum of the delay experienced by the data packet on the way from the sender to the receiver and the delay experienced by the ACK packet sent from the receiver to the sender. Each of these delays comprise of the time to forward the packet over h hops, where each forwarding requires a queuing delay t_q and transmission delays t_{data}, and t_{ACK}, respectively. The parameters involved in the estimation of the 4-hop propagation delay are given in Table 1. Using the measured RTT, we can write:

$$RTT = h\left(t_q + t_{data}\right) + h\left(t_q + t_{ACK}\right)$$

Solving for t_q while using $t_{data} = s_{data}/b$ and $t_{ACK} = s_{ACK}/b$, we derive the average queuing delay as:

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

Subsequently, we can estimate the 4-hop propagation delay of the TCP data packet:

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

Note that this estimation requires that the TCP sender knows about the number of hops on the network path to the receiver and the bandwidth of the wireless network interface. Since this information is available on the routing and MAC layers at the source node, no extra overhead is required. A more accurate estimation of the 4-hop propagation delay may be achieved if the clocks of both TCP entities are synchronized. If available, this clock synchronization can be obtained using the Global Positioning System (GPS) [10]. In Section 5.1, we discuss how synchronized clocks affect the performance of TCP-AP.

4.4 The Adaptive Pacing Scheme

Since the computation of the adaptive transmission rate should account for both the current contention on the network path and the spatial reuse constraint, we incorporate cov_{RTT} and FHD in the transmission rate formula. Recall that a rate of $R_{max}=1/FHD$ specifies an upper bound for the achievable goodput under optimal conditions, i.e. with perfect scheduling and no contention. In order to adaptively throttle the transmission rate R according to the current degree of contention, we use cov_{RTT} as additional decay factor:

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

The coefficient of variation quantifies the percentage of sample deviation from the mean. However, since we want to quantify the size of the spectrum in which the samples fluctuate around the mean, we double the value cov_{RTT} in the rate formula.

Note that in favor of a stable transmission rate, we have to average the measured 4-hop propagation delay samples and

```
proc recv():
  comment : procedure called upon ACK receipt
  Variables :
      InterPacketDelay: time between successive
                        packet transmissions

  for each received ACK do
    FHD ← 2 ( RTT/h + (s_data − s_ACK)/b )
    FHD_new ← α · FHD_old + (1−α) · FHD
    calculate cov_RTT over most recent N RTT samples
    InterPacketDelay ← FHD_new · (1 + 2cov_RTT)
  done

proc pacing_timeout(InterPacketDelay):
  comment : procedure called every InterPacketDelay time
  Variables :
      seqno       : current TCP sequence number
      highestACK  : sequence number of last ACK received
      awnd        : receiver advertised window size
      cwnd        : congestion window size

  if seqno ≤ highestACK + min(awnd, cwnd) then
    send new packet
  else
    stay idle
  fi
```

Figure 8: Congestion control of TCP-AP

employ a reasonable history size N for the computation of the coefficient of variation. Recall from Section 4.2 that N denotes the number of the most recent samples used for determining cov_{RTT}. For averaging the 4-hop propagation delay samples, we use the exponentially weighted moving average (EWMA) with averaging weight α.

That is:

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

Putting it together, the algorithm given in Figure 8 specifies the proposed rate-based congestion control executed at the TCP sender. Note that we preserve the retransmission strategy of regular TCP for handling packet losses.

We would like to point out that the computational cost for deriving cov_{RTT} could be optimized in a deployed implementation of TCP-AP. In particular, the standard deviation could be approximated by the mean absolute deviation without requiring the calculation of a square root as proposed in [12] for quantifying the variability of RTT's.

Obviously, there is a trade-off between the stability of the transmission rate and the responsiveness of the algorithm. This trade-off is controlled by the weight α and the history size N. In the following, we experimentally derive appropriate values for these two parameters.

4.5 Parameter Tuning and Responsiveness

As responsiveness we denote how quickly the congestion control algorithm adapts to changing network conditions such as additional flows competing for the bandwidth. We analyze the impact of the averaging weight α and the history size N on goodput, fairness and responsiveness of TCP-AP. Obviously, the responsiveness of TCP-AP improves for smaller values of α and

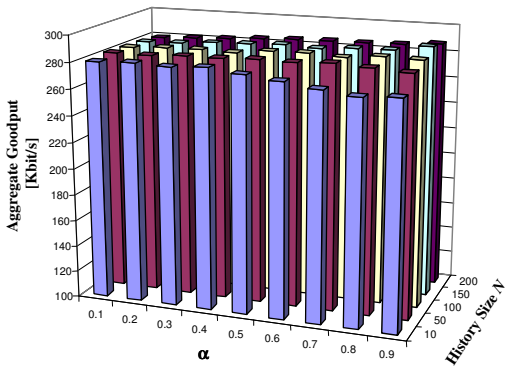

Figure 9: Aggregate goodput for different parameter settings in the cross topology

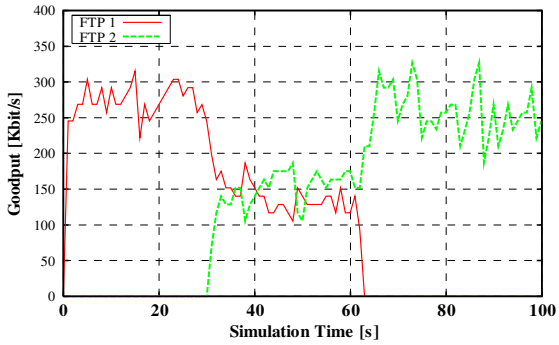

Figure 10: Responsiveness of TCP-AP

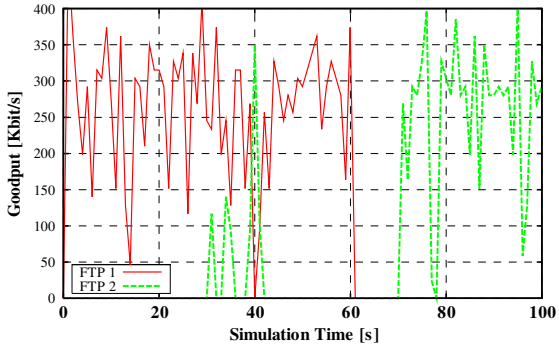

Figure 11: Responsiveness of TCP NewReno

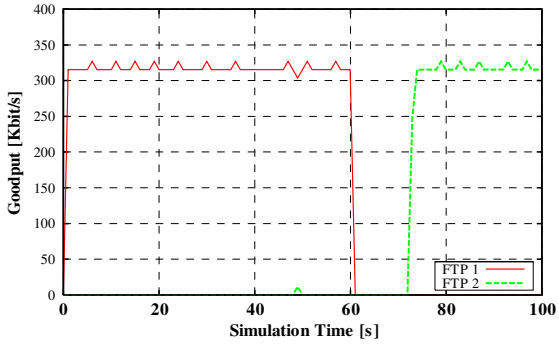

Figure 12: Responsiveness of TCP NewReno with optimal CWL

N, since the most recently measured RTT samples gain more influence on the computed transmission rate. However, we observe that an appropriate value of *N* has a more significant influence on responsiveness than α. This can be attributed to the fact that in the computation of cov_{RTT}, the mean is determined using a simple moving average. Since the last measured sample has only a weight of 1/*N*, the average value cannot respond as quickly to changing conditions as an EWMA with small averaging weight.

To find a parameter set which provides good performance in terms of goodput and fairness while achieving appropriate responsiveness, we conduct a simulation study using the cross topology of Figure 1. In this scenario, we define two TCP-AP flows instead of one fixed rate TCP and one UDP flow. Figure 9 plots the aggregate goodput, i.e. the sum of the goodput achieved by both flows for different values of α and *N*. We find that the highest aggregate goodput is achieved for large values of α and *N*. For history sizes of 50 or more samples, the aggregate goodput increases slightly with increasing α. However, for a history size of 10 samples, the aggregate goodput decreases with increasing averaging weight. As a reasonable parameter set, which provides good responsiveness by using a possibly small history size, combined with adequate goodput, we choose *N*=50 and α=0.7. This parameter set achieves a superior compromise between responsiveness and goodput not only in the cross topology but also in the symmetric parallel chains topology depicted in Figure 2. Due to space limitations, we omit the corresponding curve.

We evaluate the fairness achieved by our approach for the different parameter sets by computing Jain's fairness index [13], which is defined as:

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

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

For all parameter sets, TCP-AP achieves a fairness index of more than 0.99. Thus, our approach perfectly shares the available bandwidth upon the two competing flows.

To illustrate the transient behavior of TCP-AP compared to TCP NewReno, Figures 10 to 12 plot the goodput versus time for the cross topology. For this simulation set, the first TCP flow runs from the beginning until T_2=60s and the second TCP flow starts at T_1=30s and runs until the end of the simulation. Figure 10 shows that TCP-AP responds quickly to changing network conditions when new flows start (T_1) as well as when flows stop (T_2). Using TCP-AP, both flows utilize the available bandwidth when there are no competing flows and share the bandwidth fairly when multiple flows compete for the bandwidth. As already observed in previous work (e.g. [9], [18]), TCP NewReno suffers from serious unfairness. From Figure 11, which shows the goodput of two TCP NewReno flows over time, we observe that the first flow occupies almost the entire bandwidth during its lifetime, even after the second flow starts. Thus, the second flow experiences multiple timeouts and increases the retransmission timeout (RTO). Such large retransmission timeouts lead to underutilization of the available bandwidth when the first flow ends at T_2. We further observe that, opposed to TCP-AP, the goodput of TCP NewReno fluctuates strongly over time. This is due to the inappropriate

Figure 13: 7-hop chain topology with a single flow

strategy of TCP NewReno for adjusting its congestion window, which results in overshooting the optimal window size rather than smoothly sensing the available bandwidth.

Similar unfairness can be observed in Figure 12 for TCP with fixed congestion window limit (CWL), as proposed in [6]. Here, the congestion window is limited to the optimal value for this topology, namely one for a 4-hop chain. We find that the first TCP flow achieves near-optimal goodput during its lifetime but does not share the bandwidth with the newly started second flow at T_1. Due to the same reasons as for TCP NewReno, the second flow does not utilize the spare bandwidth, which becomes available when the first flow stops.

4.6 Competing TCP-AP and TCP NewReno Flows

Since TCP-AP applies rate-based transmission only within the TCP congestion window, the number of packets in flight can never exceed what the window allows. Consequently, TCP-AP is TCP-friendly by definition, since TCP-AP cannot transmit at a rate higher than a standard TCP connection operating along the same path would achieve. On the other side, in multihop wireless networks in which a TCP-AP flow competes with a TCP NewReno flow, it is obvious that the TCP NewReno flow will obtain most of the available bandwidth due to its aggressive window control. However, as observed in Figure 11, this is a typical behavior of TCP NewReno, even when competing with other TCP NewReno flows. In fact, the aggressive behavior of TCP NewReno would result in the occupancy of most of the available bandwidth when competing with any transport protocol implementing proactive congestion control.

Other approaches such as the neighborhood RED scheme [18], which force TCP NewReno to decrease its aggressive transmission by purposely dropping packets on intermediate nodes may yield improved fairness in multihop wireless networks with mixed TCP flows. Since the deficiencies of TCP NewReno for multihop wireless networks are well known, such interactions between NewReno and non-NewReno TCP variants play for multihop wireless networks a less significant role than in the Internet.

5. COMPARATIVE PERFORMANCE STUDY

To illustrate the effectiveness of TCP-AP, we present a comprehensive performance study of TCP-AP versus TCP NewReno with and without dynamic delayed ACK [1]. The dynamic delayed ACK approach extends the basic delayed ACK option for TCP [4] by dynamically setting the parameter *d* which represents the number of packets received by the TCP receiver before an acknowledgment is generated. The parameter *d* gradually increases from one to four based on the sequence numbers of the received TCP packets. Although the delayed ACK option decreases the frequency of RTT samples measurable by the TCP sender, this reduction of feedback has no significant impact on the effectiveness of the adaptive pacing scheme in TCP-AP.

5.1 FTP-Like Data Transfer

In the first set of experiments, we consider scenarios with FTP-like data transfer in different network topologies. That is, the TCP sender transmits packets continuously, representing a large file transfer.

5.1.1 Chain Topology

As a first scenario, we consider an equally spaced chain comprising of h+1 nodes (h hops) with a single flow as depicted in Figure 13. TCP packets travel along the chain from the leftmost node (i.e., the sender) to the rightmost node (i.e., the receiver). Figure 14 plots the goodput of the examined TCP variants for varying hop number. We observe that TCP-AP outperforms TCP NewReno for all hops by up to 84%. In fact, TCP-AP even outperforms TCP NewReno with delayed ACK for $h > 8$, whereas TCP-AP with delayed ACK achieves 7% to 17% more goodput than TCP-AP.

In order to determine the optimum goodput achievable in an h-hop chain, we conduct a further experiment, in which the examined TCP variants are provided with global knowledge. That is, we utilize synchronized clocks for both TCP sender and receiver for TCP-AP to determine more accurate estimates of the 4-hop delays. For TCP NewReno, we set the optimal congestion window limit size (CWL) for each number of hops according to [6]. We compare these TCP variants to a paced UDP flow with an optimized transmission rate for each number of hops. Similar to TCP, the UDP packet size is set to 1460 bytes, whereas the

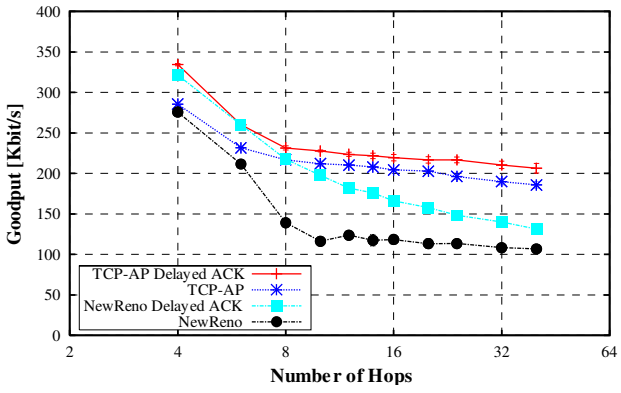

Figure 14: TCP goodput without global knowledge

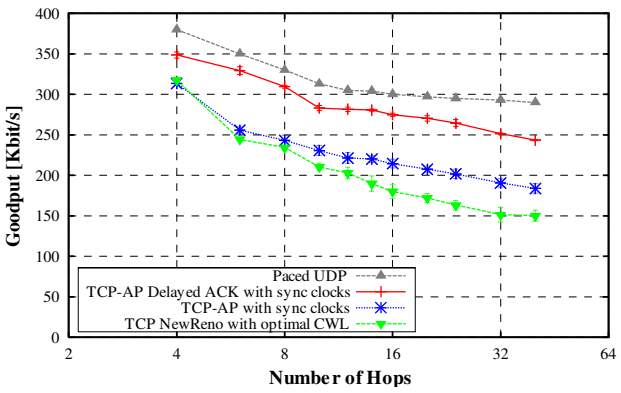

Figure 15: TCP goodput with global knowledge

overhead caused by TCP packet retransmissions and acknowledgments is neglected. We use paced UDP as an upper bound for the highest goodput achievable in such scenario. Figure 15 shows that TCP-AP with synchronized clocks achieves up to 27% more goodput than TCP NewReno with optimal CWL, whereas applying delayed ACK yields 9% to 36% further improvement for TCP-AP. In fact, TCP-AP with delayed ACK achieves only 5% to 16% less goodput than the simulative upper bound given by paced UDP. Note that delayed ACK cannot be applied for TCP NewReno with optimal CWL. This is because for $h \leq 10$, CWL is less than four, and thus, smaller than the parameter d used for delayed ACK. Therefore, the TCP receiver would experience a lack of TCP packets and time out each time the TCP sender transmits a total of CWL packets, causing severe goodput decrease.

5.1.2 Symmetric Parallel Chains Topology

In this scenario, we use the symmetric parallel chains topology illustrated in Figure 2, where we consider one TCP flow for each chain. Figure 16 plots the goodput achieved by the two individual flows as well as the aggregate goodput for both flows. We find that for TCP NewReno, the first flow utilizes most of the available bandwidth at the cost of starving the second flow. In fact, we notice that the second flow only achieves about 10% of the overall goodput. TCP NewReno with delayed ACK improves the goodput of both flows, though, the fairness remains insufficient. Opposed to TCP NewReno, TCP-AP with and without delayed ACK achieve optimal fairness. In fact, TCP-AP even achieves a better utilization of the available bandwidth, obtaining more aggregate goodput than TCP NewReno. This is because TCP-AP minimizes both inter-node and inter-chain contention by adapting its transmission rate according to the fluctuation of the RTT samples. Figure 17 shows the coefficient of variation of TCP-AP for both FTP flows. We notice that it fluctuates within the same range for both flows, namely between zero and one, according to the level of network contention at a specific time. Note that mechanisms that require a direct communication between nodes in order to improve TCP fairness would not yield any gain in scenarios like the symmetric parallel chains topology. This is because the distance between the chains does not allow inter-chain communication.

5.1.3 Asymmetric Parallel Chains Topology

To evaluate the impact of the chain length on TCP fairness, we consider an asymmetric parallel chains topology. Extending the topology in Figure 2, we expand the upper chain by three hops in each direction. Figure 18 plots the goodput results for this scenario. We observe that for both TCP NewReno and TCP NewReno with delayed ACK, FTP 1 running over the shorter chain obtains almost all of the available bandwidth, letting FTP 2 nearly completely starve. TCP-AP achieves significantly better fairness than TCP NewReno, although the available bandwidth is not shared equally between both flows. This has mainly two reasons. (1) The maximum achievable goodput in a chain topology depends strongly on the number of hops, as previously observed in Figure 14. The goodput decreases with increasing number of hops. Thus, FTP 1 achieves more goodput than FTP 2. (2) The TCP sender experiences more fluctuation of the RTT samples due to multiple hidden terminals for longer chains. This results in larger values for the coefficient of variation and reduces the transmission rate. Since delaying ACKs reduces network

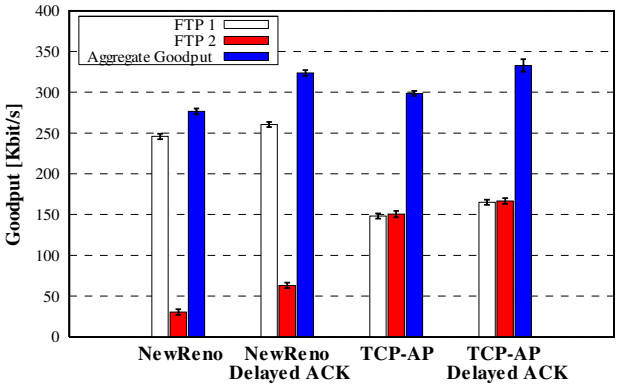

Figure 16: Goodput and fairness for the symmetric parallel chain

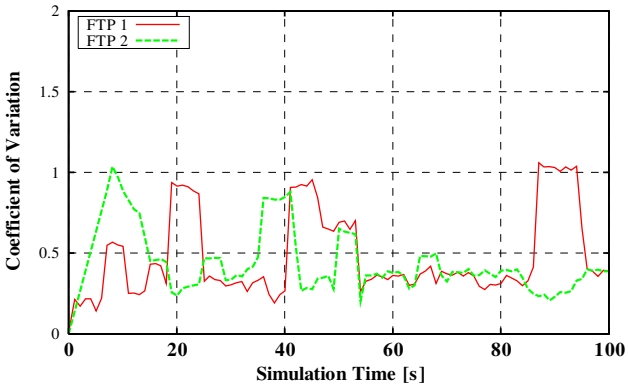

Figure 17: Transient fluctuation of the coefficient of variation in TCP-AP

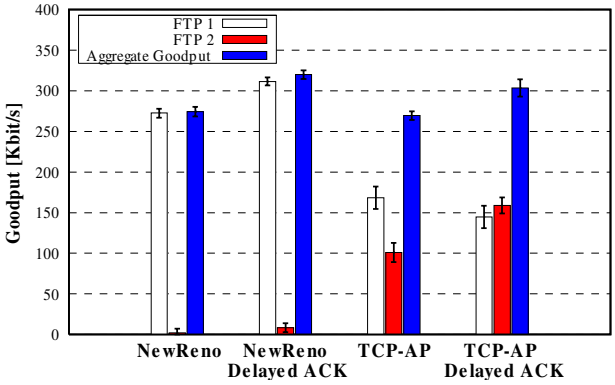

Figure 18: Goodput and fairness in asymmetric parallel chains topology

contention, the fluctuation of the RTT samples decreases for TCP-AP with delayed ACK. As we can see in Figure 18, this leads to improved bandwidth utilization for FTP 2, letting both FTP flows achieve similar goodput. Considering the aggregate goodput, we notice that the TCP NewReno variants achieve slightly more aggregate goodput than TCP-AP. This is due to the known trade-off between aggregate goodput and fairness caused by the absence of optimal scheduling of the IEEE 802.11 MAC protocol. This problem was discussed in [15] and [18], where [18] reported up to

42% less aggregate goodput for scenarios in which neighborhood RED achieves optimal fairness.

5.1.4 Grid Topology

We consider a highly congested grid topology with six FTP flows as shown in Figure 19. Figure 20 plots the goodput results for this scenario. We observe that for TCP NewReno, FTP 1 and FTP 6 are basically the only active flows, causing an almost total starvation of the remaining four flows. TCP NewReno with delayed ACK improves the fairness slightly, though FTP 1 and FTP 6 still obtain more than 12 times the goodput of the remaining four flows. Opposed to TCP NewReno, TCP-AP significantly improves fairness. In fact, TCP-AP lets FTP 1 and FTP 6 sacrifice some of their goodput for the benefit of the remaining flows. We notice that both FTP 1 and FTP 6 still get higher goodput than the remaining flows, both for TCP-AP and TCP-AP with delayed ACK. Such effect depends on the relative position of the FTP flows within the grid. That is, FTP 1 experiences less contention than FTP 2 and FTP 3 due to its relatively far position to the TCP senders of FTP 4 to FTP 6. Thus, the probability that a packet of FTP 3 collides with packets of FTP 4 to FTP 6 is higher than the probability that a packet of FTP 1 collides with packets of FTP 4 to FTP 6. The same applies to FTP 6, whose position corresponds to the position of FTP 1. This leads to the intuition that FTP 2 and FTP 5 should achieve more goodput than FTP 3 and FTP 4, respectively. However, contrary to intuition, the simulation results show that FTP 2 and FTP 5 achieve slightly less goodput than FTP 3 and FTP 4. This is due to the fact that the nodes transferring FTP 2 receive all RTS/CTS packets of the neighboring parallel chains transferring FTP 1 and FTP 3. Note that the chain transferring FTP 1 is out of the transmission range of nodes transferring FTP 3 and vice versa. Thus, nodes transferring FTP 2 receive more RTS/CTS signals from the neighboring chains than nodes transferring FTP 1 and FTP 3, respectively. Hence, they throttle their transmission in favor of the neighboring flows. In a symmetric grid scenario not shown, where the relative positions are identical for each flow, TCP-AP achieves perfect fairness between the competing flows.

5.1.5 Random Topology

In order to get intuition on the performance of the examined TCP variants in more realistic scenarios, we consider a random topology of 120 nodes uniformly distributed in an area A = 2500m x 1000m. According to [3], in this setup, all nodes in the network can communicate with each other over one or more hops with probability P=99.9%. Thus, almost surely there exists at least one path between any TCP sender/receiver pair. We choose ten TCP sender/receiver pairs randomly from the set of nodes, each running an FTP file transfer of unlimited size. Figure 21 plots goodput results for this scenario. Consistent with the previous results, we find that the fairness of TCP NewReno is very bad. In fact, TCP NewReno lets five FTP flows almost completely starve, while FTP 4 occupies most of the available bandwidth. TCP NewReno with delayed ACK improves the aggregate goodput, though FTP 4 still obtains most of the available bandwidth at cost of the remaining flows. Opposed to the TCP NewReno variants, TCP-AP achieves significantly better fairness between the FTP flows, letting FTP 4 sacrifice a fraction of its goodput for the benefit of the remaining flows. Particularly, none of the ten FTP flows experiences starvation. The reason for the goodput difference between FTP 4 and the remaining flows is that,

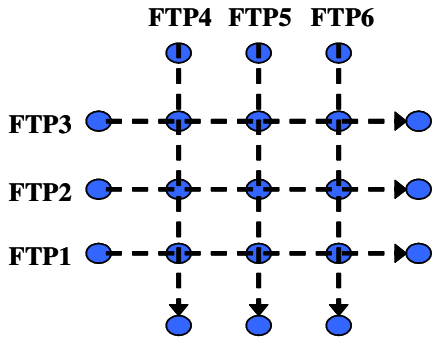

Figure 19: Grid topology

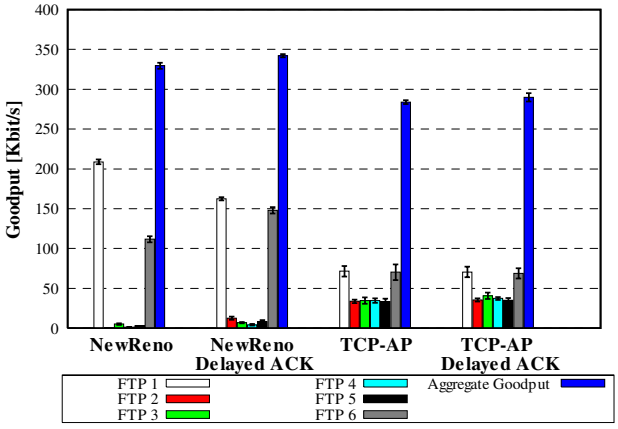

Figure 20: Goodput and fairness in grid topology

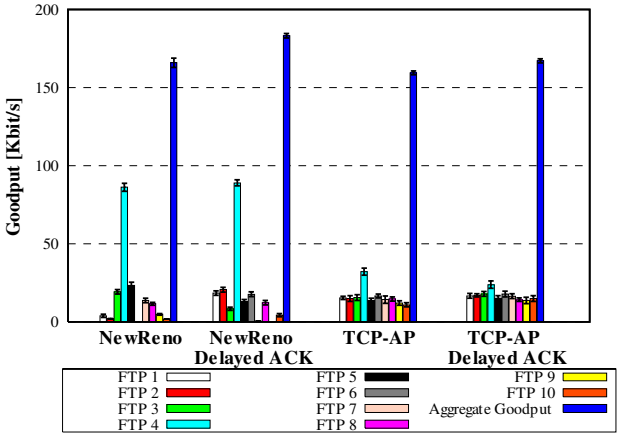

Figure 21: Goodput and fairness in random topology

opposed to the other FTP flows, the TCP entities of FTP 4 are only two hops away from each other, which translates to a higher available bandwidth for the flow, as already discussed in the asymmetric parallel chain scenario. We further observe that TCP-AP with delayed ACK results in even more goodput decrease for FTP 4, and thus, achieves better fairness. This behavior is also consistent with the results of the scenario with the asymmetric parallel chains topology. Note that although TCP-AP achieves considerably better fairness than TCP NewReno, it only sacrifices

6% aggregate goodput due to the trade-off between aggregate goodput and fairness.

5.2 Data Transfer with Variable Length Flows

In the second set of experiments, we consider variable length flows, where the TCP sender transmits small files with variable pause times between successive file transfers. Following [14], we assume that the file sizes are Pareto distributed with mean 30 Kbytes and shape factor $\beta = 1.5$, whereas pause times between successive file transfers are exponentially distributed.

5.2.1 Symmetric Parallel Chains Scenario

We re-iterate the scenario of the symmetric parallel chains used in Section 5.1, though, we substitute the continuous FTP flows by flows of variable length. In order to evaluate the impact of the pause times on the performance of the examined TCP variants, we vary the mean of the exponentially distributed pause times from 0.1 to 1 second. Additionally, we consider zero pause times. Obviously, the goodput for zero pause time corresponds to the results for continuous FTP flows shown in Figure 16. As measures of interest, we consider the fairness index and the achieved aggregate goodput averaged per flows. The latter measure is given by the achieved averaged goodput summed up for all flows. Note that we do not consider the pause times in the calculation of the averaged goodput. That is, we determine the amount of unique data received by the TCP receiver and divide it by the actual time needed for transmitting this data.

In Figure 22 we observe that the aggregate averaged goodput of TCP-AP increases with increasing mean for the pause times, whereas the aggregate averaged goodput of TCP NewReno does not change. This observation can be explained as follows: As the pause times between file transfers increase, the probability that both flows transmit packets simultaneously decreases. Thus, each flow gets a higher chance for taking advantage of the entire available bandwidth when the other flow is not transmitting. Moreover, the probability for packet collisions also decreases for increasing pause times. Since TCP-AP responds quickly to changing traffic conditions such as the starting and ending of other flows, it can take advantage of such pause times when the other flow is idle, and thus, achieves more goodput. On the other side, TCP NewReno does not take advantage of the pause times since it cannot utilize the available bandwidth fast enough, as we showed in Section 4.5. This effect even has a stronger impact when transferring small files, as it is the case in this scenario, since TCP should be able to utilize the available bandwidth quickly in order to get benefit from the pause times, when the other flow is idle.

Regarding fairness, we observe in Figure 23 that both TCP-AP and TCP NewReno achieve near-optimal fairness for all non-zero pause times. While this is a typical behavior for TCP-AP, it is untypical for TCP NewReno. In fact, TCP NewReno achieves a suboptimal fairness of about 0.64 for zero pause times. That is, increasing the mean for the pause times increases the chance for a starved flow to gain the entire bandwidth at the cost of the other flow. Thus, both flows use the entire bandwidth almost alternating which results in good fairness on average. However, opposed to

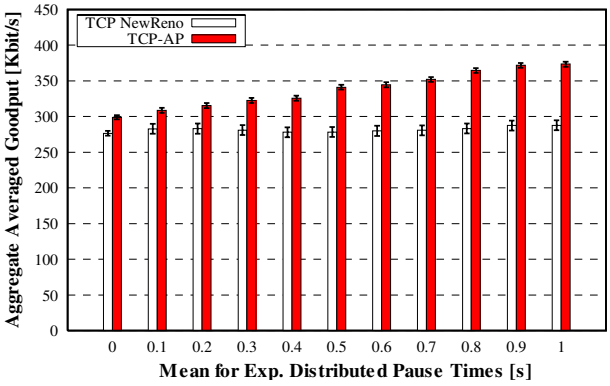

Figure 22: Aggregate averaged goodput of TCP NewReno and TCP-AP vs. mean for pause times

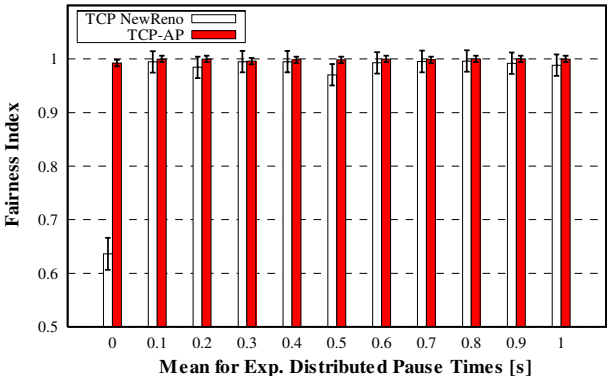

Figure 23: Fairness of TCP NewReno and TCP-AP vs. mean for pause times

TCP NewReno, the fairness of TCP-AP is also optimal when both flows transmit continuously without pause times. A further simulation shows that applying delayed ACK for both TCP NewReno and TCP-AP results in an increase in the aggregate averaged goodput while providing the same qualitative results as without delayed ACK.

5.2.2 Random Scenario

Once again, we consider a random scenario with the same settings as in Section 5.1, except that we simulate five TCP flows with variable length instead of ten continuous FTP flows. For this scenario, we choose a fixed mean of 1 second for the exponentially distributed pause times between successive file transfers. Figure 24 shows the results for this simulation. We find that the TCP-AP variants significantly outperform the NewReno variants, both in terms of aggregate averaged goodput and fairness. Specifically, TCP-AP achieves about 91% more aggregate averaged goodput than TCP NewReno, while TCP-AP with delayed ACK achieves about 58% more aggregate averaged goodput than TCP NewReno with delayed ACK. Looking at the averaged goodput of each flow, we observe that TCP-AP provides better fairness among the flows than TCP NewReno. We also notice that applying delayed ACK for both TCP NewReno and TCP-AP results in an improvement both in terms of aggregate averaged goodput and fairness.

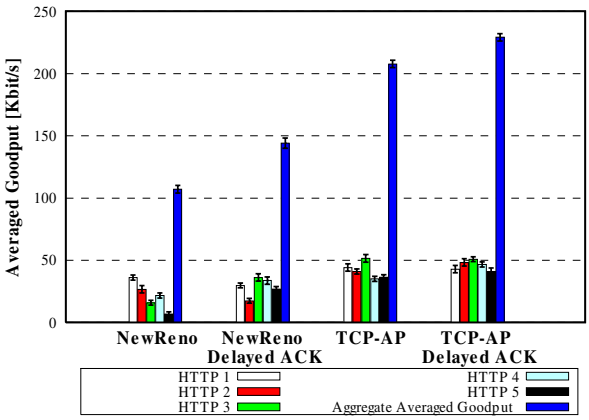

Figure 24: Averaged goodput and fairness in random topology

6. CONCLUSIONS

We proposed a novel congestion control algorithm for TCP over multihop wireless networks denoted as TCP with Adaptive Pacing (TCP-AP). TCP-AP implements rate-based scheduling within TCP's congestion window in order to avoid bursty packet transmissions. The key feature of the proposed algorithm is the quantification of incipient congestion by measuring the fluctuation of round trip time samples using the coefficient of variation. Based on this measure for contention on the network path, as well as the estimation of 4-hop propagation delays, TCP-AP adaptively calculates the appropriate rate for pacing the transmission. Since TCP-AP relies solely on end-to-end measurements of round trip times and requires no modifications on the routing layer or the link layer, TCP-AP is easily deployable.

In a comprehensive simulation study using ns-2 [8], we showed that TCP-AP achieves up to 84% more goodput than TCP NewReno and provides excellent fairness in almost all scenarios. In particular, TCP-AP provides fair sharing of the available bandwidth, even when competing flows are not within each other's transmission range, but within each other's interference range, since the proposed algorithm relies on the end-to-end measurement of the interference experienced by a TCP connection.

In future work, we are examining the performance of TCP-AP over routing protocols, which use different routing metrics such as energy efficiency or expected transmission count (ETX) [7]. Furthermore, we would like to implement TCP-AP for an IEEE 802.11 testbed in order to evaluate and optimize its performance in real multihop wireless networks.

7. REFERENCES

[1] E. Altman and T. Jimenez, Novel Delayed ACK Techniques for Improving TCP Performance in Multihop Wireless Networks, *Proc. Personal Wireless Communications (PWC 03)*, Venice, Italy, 2003.

[2] A. Aggrawal, S. Savage, and T. Anderson, Understanding the Performance of TCP Pacing, *Proc. IEEE INFOCOM 00*, Tel Aviv, Israel, 2000.

[3] C. Bettstetter, On the Minimum Node Degree and Connectivity of a Wireless Multihop Network, *Proc. ACM MobiHoc 02*, Lausanne, Switzerland, 2002.

[4] R. Braden, *Requirements for Internet Hosts - Communication Layers*, IETF RFC 1122, 1989.

[5] K. Chen, K. Nahrstedt, and N. Vaidya, The Utility of Explicit Rate-Based Flow Control in Mobile Ad Hoc Networks, *Proc. IEEE Wireless Communications and Networking Conference (WCNC 04)*, Atlanta, GA, 2004.

[6] K. Chen, Y. Xue, S. Shah, and K. Nahrstedt, Understanding Bandwidth-Delay Product in Mobile Ad Hoc Networks, *Computer Communications Journal*, **27**, 923-934, 2004.

[7] D. De Couto, D. Aguayo, J. Bicket, and R. Morris, A High-Throughput Path Metric for Multi-Hop Wireless Routing, *Proc. ACM MOBICOM 03*, San Diego CA, 2003.

[8] K. Fall and K. Varadhan (Ed.), The ns-2 Manual, *Technical Report*, The VINT Project, UC Berkeley, LBL, and Xerox PARC, 2003.

[9] 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 03*, San Francisco CA, 2003.

[10] I. A. Getting, The Global Positioning System, *IEEE Spectrum 30,* December 1993.

[11] H. Holland and N. Vaidya, Analysis of TCP Performance over Mobile Ad Hoc Networks, *Proc. ACM MOBICOM 99*, Seattle, WA, 1999.

[12] V. Jacobson, Congestion Avoidance and Control, *Proc. ACM SIGCOMM 88*, Stanford, CA, September, 1988.

[13] R. Jain, D. Chiu, and W. Hawe, A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Systems, *DEC Technical Report DEC-TR-301*, 1984.

[14] Y. Joo, V. Ribeiro, A. Feldmann, A.C. Gilbert, and W. Willinger, TCP Traffic Dynamics and Networks Performance: A Lesson in Workload Modeling, Flow Control, and Trace-driven Simulation, *ACM SIGCOMM Computer Communication Review*, **31**, 2001.

[15] H. Luo, S. Lu, and V. Bharghavan, A New Model for Packet Scheduling in Multihop Wireless Networks, *Proc. ACM MOBICOM 00*, Boston, MA, 2000.

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

[17] 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.

[18] K. Xu, M. Gerla, L. Qi, and Y. Shu, Enhancing TCP Fairness in Ad Hoc Wireless Networks using Neighborhood RED, *Proc. ACM MOBICOM 03*, San Diego CA, 2003.

[19] S. Xu and T. Saadawi, Performance Evaluation of TCP Algorithms in Multi-Hop Wireless Packet Networks, *Wireless Communication and Mobile Computing*, **2**, 85-100, 2002.

[20] X. Yu, Improving TCP Performance over Mobile Ad Hoc Networks by Exploiting Cross-Layer Information Awareness, *Proc. ACM MOBICOM 04,* Philadelphia, PA, 2004.