Improving TCP Performance for Multihop Wireless Networks*

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

Mary K. Vernon
University of Wisconsin - Madison
Department of Computer Sciences
1210 West Dayton Street
Madison, WI 53706
http://www.cs.wisc.edu/~vernon/

Abstract

In this paper, we present a comprehensive performance evaluation of TCP NewReno and TCP Vegas with and without ACK thinning for static multihop wireless IEEE 802.11 networks. Opposed to previous studies, we consider not only IEEE 802.11 operating in ad hoc mode with 2 Mbit/s bandwidth, but also with 5.5 Mbit/s and 11 Mbit/s bandwidths. Simulation results using ns-2 show that TCP Vegas achieves between 15% and 83% more goodput and between 57% and 99% fewer packet retransmissions than TCP NewReno. Considering fairness among multiple TCP flows, we show that using TCP Vegas results in between 21% and 95% fairness improvement compared to TCP NewReno. The reduced amount of packet retransmissions of TCP Vegas also leads to significant savings of energy consumption. The paper gives insight on the particular reasons for such performance advantages of TCP Vegas in comparison to TCP NewReno.

1 Introduction

Numerous mobile applications for ad hoc networked PDAs and laptops over IEEE 802.11 wireless technology require a reliable transport protocol like TCP. Such multihop wireless networks possess several properties, which are different to the wired Internet for which widely deployed TCP implementations like TCP Reno and TCP NewReno have been optimized. In particular, the wireless channel is a scarce resource shared among nodes within their radio range. Thus, TCP segments may not only be lost due to buffer overflow, but also due to link-layer contention caused by hidden terminals [5]. A hidden terminal is a potential sending node in the receiver's neighborhood, which cannot detect the sender and may disrupt an ongoing transmission of a TCP segment. In fact, as mentioned in [5] and further verified by our simulation, for multihop wireless networks using IEEE 802.11 most losses experienced by TCP are due to packet drops at the link layer incurred by hidden terminal effects and not due to buffer overflow.

* The research in this paper was partially supported by the German Research Council (DFG) under Grant Li-645/12-2 and by the U.S. National Science Foundation under Grant ANI-0117810.

TCP Vegas was introduced by Brakmo et al. [3] as an alternative TCP variant with innovative features for congestion control and packet retransmission. Opposed to the widely deployed transport protocol TCP NewReno, TCP Vegas tries to sense incipient congestion by monitoring the current throughput. It has been shown that for the Internet, TCP Vegas achieves considerably higher throughput and fewer losses than TCP Reno [3], [10]. However, little work has been done to investigate the performance, fairness, and energy efficiency of TCP Vegas in multi-hop wireless networks over IEEE 802.11.

Recently, commercial products based on the IEEE 802.11g standard have become available providing bandwidths up to 108 Mbit/s, [8]. As a consequence, future ad hoc networked PDAs and laptops over IEEE 802.11 wireless technology may well operate with bandwidths higher than 2 Mbit/s. Thus, opposed to previous studies [5], [14], we consider not only IEEE 802.11 wireless network technology operating in ad hoc mode with 2 Mbit/s bandwidth, but also with 5.5 Mbit/s and with 11 Mbit/s bandwidth.

In this paper, we present a comprehensive performance evaluation of TCP NewReno and TCP Vegas for static multihop wireless IEEE 802.11 networks. We consider an *h*-hop chain without cross traffic, a grid topology with six competing flows and a random topology with ten concurrent flows, over IEEE 802.11 wireless network technology. To get intuition on the optimum achievable goodput over an IEEE 802.11 network with a given bandwidth, we consider an optimally paced UDP protocol, which exploits knowledge of the optimal packet transmission rate in a chain topology. Simulation results obtained by ns-2 [4] show that TCP Vegas clearly outperforms TCP NewReno in static multihop wireless networks. In fact, TCP Vegas achieves between 15% and 83% more goodput and between 57% and 99% fewer packet retransmissions than TCP NewReno. Furthermore, the more conservative window control of TCP Vegas yields between 21% and 95% better fairness than TCP NewReno in multi-flow environments. The reduced amount of packet retransmissions of TCP Vegas also results in significant savings of energy consumption. We further show that thinning TCP acknowledgements, recently proposed for TCP NewReno over IEEE 802.11 [1], improves both

Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)
0-7695-2282-3/05 $20.00 © 2005 IEEE

fairness and goodput of TCP Vegas and TCP NewReno for increasing bandwidth, letting TCP Vegas with ACK thinning achieve the best overall results among all examined TCP variants.

The remainder of this paper is organized as follows. Section 2 summarizes related work on TCP for multihop wireless networks. In Section 3, we recall the congestion control algorithm of TCP Vegas as well as thinning acknowledgements for improving TCP performance. A comprehensive performance study of TCP Vegas versus TCP NewReno with and without thinning acknowledgements is presented in Section 4. Finally, concluding remarks are given.

2 Related Work

Several efforts for improving the performance of TCP in mobile ad hoc networks based on IEEE 802.11 technology have recently been reported. Fu et al. [5] pointed out the hidden terminal problem in such networks and proposed two enhancements: adaptive pacing on the link-layer and link-layer RED. Using simulation with ns-2 [4], they showed that these link-layer enhancements improve throughput by 5% to 30%. They considered TCP NewReno over IEEE 802.11 with 2 Mbit/s bandwidth. Furthermore, they commented that TCP NewReno achieves better performance than TCP Vegas for an h-hop chain topology with $h \geq 9$. We are presenting a comprehensive performance study of TCP Vegas versus TCP NewReno. Consistent with [5], we observe that for the h-hop chain the optimum TCP window size is given by $h/4$. Opposed to [5], our simulation results evidently show that TCP Vegas with parameters $\alpha = \beta = 2$ outperforms TCP NewReno in static multihop wireless networks.

Altman and Jiménez [1] proposed an adaptive scheme for delaying TCP acknowledgements (subsequently denoted as ACK thinning) in order to improve TCP throughput in multihop wireless networks. They also considered TCP NewReno over IEEE 802.11 with 2 Mbit/s bandwidth. Using simulation with ns-2, they showed that for an h-hop chain, ACK thinning yields around 50% more throughput. Building upon their results, we are also considering ACK thinning, though, not only for TCP NewReno, but also for TCP Vegas. Beyond [1], [5], we are comparing TCP NewReno and TCP Vegas with and without ACK thinning against a paced UDP in order to get intuition how an optimum transport protocol over IEEE 802.11 may perform. Furthermore, we are not only considering 2 Mbit/s bandwidth, but also 5.5 Mbit/s and 11 Mbit/s bandwidths.

In [15], Saadawi and Xu investigated the performance of TCP Vegas in multihop wireless networks among four further TCP variants, reporting 15% to 20% more goodput for TCP Vegas. However, the results of the study were based on a chain topology with a maximum of 7 hops and a single TCP flow. Opposed to [15], we investigate the performance of TCP Vegas in more complex, multi-flow environments while regarding a further key performance aspect, namely TCP fairness. Moreover, using simulation, we determine the best values for the Vegas specific parameters α and β in order to achieve the best performance for TCP Vegas. Opposed to [15], our simulation study shows that TCP Vegas achieves up to 83% more goodput than TCP NewReno.

Several authors introduced TCP enhancements for coping with mobility in ad hoc wireless networks over IEEE 802.11. Holland and Vaidya [7] introduced explicit link failure notification (ELFN) as a feedback mechanism from the network in order to help TCP to distinguish between congestion losses and losses induced by link failures due to mobility. To help TCP coping with mobility, Wang and Zhang [13] proposed detection and out-of-order response (DOOR) as a new way to make TCP adapt to frequent route changes without relaying on feedback from the network. We focus on TCP performance in static wireless networks instead, though, our results may well be utilized together with the findings of either [7] or [13] in order to optimize TCP performance in mobile ad hoc networks based on IEEE 802.11.

Numerous performance studies of TCP Vegas have been reported for the Internet e.g., [3], [10]. Furthermore, several analytical performance models for TCP Vegas have been introduced e.g., [10], [12]. Our simulation study considers multihop IEEE 802.11 wireless networks possessing substantially different properties than the wired Internet, though, confirms the result that TCP Vegas outperforms TCP NewReno both in terms of goodput and percentage of packet losses. To get more intuition on the performance of TCP Vegas with and without ACK thinning over IEEE 802.11 multihop wireless networks, it will be helpful to extend one of these analytical performance models.

3 Background

3.1 TCP Vegas

We assume that the reader is familiar with basic mechanisms of TCP such as slow start and congestion avoidance algorithms, the two methods for loss detection: duplicate ACKs and timeouts, etc. Currently, Reno (and NewReno) constitute the most widely known and deployed congestion control algorithm for TCP. While TCP NewReno has proven to be well suited for the Internet, TCP NewReno incorporates a quite aggressive method for predicting the available bandwidth by provoking packet losses. TCP Vegas constitute an alternative TCP variant with innovative features for congestion control and packet retransmission. A source in TCP Vegas anticipates the onset of congestion by monitoring the difference between the throughput it is expecting to see and the actually realizing throughput. Compared to TCP NewReno, TCP Vegas includes a modified retransmission strategy as well as new methods for congestion detection during slow start and congestion avoidance.

To keep the paper self-contained, the following outlines the main innovation of TCP Vegas with respect to NewReno. The congestion control mechanism of TCP NewReno uses packet loss as indication of congestion.

Table 1: Base parameter setting for TCP NewReno and TCP Vegas

Variable	Meaning	Value
W_{max}	Maximum window advertised by the receiver	$W_{max} = 64$
W_{init}	Initial window used in slow start and after a timeout	$W_{init} = 1$
α	Vegas throughput threshold measured in packets	$\alpha = 2$
γ	Vegas thresholds measured for exiting slow start	$\gamma = \alpha$

Thus, TCP NewReno cannot detect the incipient signs of network congestion before losses occur and, thus, cannot prevent losses. TCP NewReno constitutes a reactive protocol, as it requires losses to determine the available bandwidth of the connection. Opposed to that, TCP Vegas tries to proactively detect congestion in its incipient stages, and subsequently reduces the TCP window in an attempt to prevent packet loss. To detect congestion, once every round trip time (*RTT*), TCP Vegas utilizes the current window size (*W*), the most recent RTT, and the minimum RTT observed so far (*baseRTT*) for computing the difference between the expected throughput, given by *W/baseRTT* and the actual throughput, estimated by *W/RTT*. That is, [3], [12]:

$$diff = (W/baseRTT - W/RTT)\ baseRTT$$

The goal of the TCP Vegas congestion avoidance algorithm lies in keeping *diff* between specific lower and upper thresholds, denoted by α and β. Throughout this paper, we set $\alpha = \beta$ because it has been shown that this parameter setting improves fairness [6]. Thus, once every RTT while not in slow start mode, TCP Vegas adjusts the window size:

$$W = \begin{cases} W+1 & if\ diff < \alpha \\ W & if\ diff = \alpha \\ W-1 & if\ diff > \alpha \end{cases}$$ [1]

Furthermore, TCP Vegas contains a more conservative slow start behavior as TCP NewReno as well as four innovative mechanisms for congestion recovery. Due to space limitations, we omit their descriptions and refer to [3] and [10]. Table 1 summarizes the parameters of TCP NewReno and Vegas considered in the simulation study presented in Section 4.

3.2 Thinning the ACK Stream in TCP

In this section, we briefly describe the dynamic ACK thinning approach introduced in [1], which aims to decrease contention on the MAC layer by thinning the ACK stream that competes with the TCP packet flow. Dynamic ACK thinning operates as follows: A parameter *d* defines the number of packets received by the TCP sink before an acknowledgment is generated. This parameter is set dynamically according to the sequence numbers of the TCP packets received and increases gradually from 1 to 4 using three defined thresholds *S1*, *S2* and *S3*. Specifically, for a received TCP packet with a sequence number *n*, $d = 1$ if $n \le S1$, $d = 2$ if $S1 \le n < S2$, $d = 3$ if $S2 \le n < S3$ and $d = 4$ if $n \ge S3$. According to [1], appropriate values for the thresholds are $S1 = 2$, $S2 = 5$ and $S3 = 9$. The reason for setting *d* dynamically according to the sequence numbers of the received packets is to prevent the TCP sink from experiencing a lack of TCP packets and freezing for a timeout of 100ms as a default value. This would be the case if the parameter *d* becomes larger than the current TCP window size. Since *d* is only being set dynamically at the beginning of the TCP connection, such case cannot be prevented if the TCP window size decreases below *d* during the remainder of the TCP connection, for instance at the initial phase of each time TCP enters slow start.

4 Comparative Performance Study

4.1 The Simulation Environment and the considered Performance Measures

To evaluate the performance of TCP NewReno and TCP Vegas with and without ACK thinning as well as paced UDP over IEEE 802.11 wireless networks, we conduct simulation experiments using the network simulator ns-2 [4]. We employ the implementations of TCP NewReno and TCP Vegas[2] as well as the MAC layer according to the IEEE 802.11 standard for wireless communication provided by ns-2. Consistent with the case in reality where the transmission range of a node would be smaller than its interference range, 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. 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 not only bandwidths of 2 Mbit/s, but also 5.5 and 11 Mbit/s. The higher bandwidths are already provided by the standard IEEE 802.11b and may well be utilized in ad hoc mode in the new standard IEEE 802.11g. Furthermore, we developed ns-2 transport agents implementing the ACK thinning mechanism and the paced UDP tailored to the considered scenarios. We assume that all TCP packets are of size 1460 bytes. For all nodes, we assume a buffer size of 50 packets. We use AODV [11] as an ad hoc routing protocol. Through our simulations we show that the behavior of AODV has a significant impact on the performance of TCP dependent on the intensity of the existing hidden terminal effects.

In all experiments, we conduct steady-state simulation starting with an initially idle system. In each run, we simulate continues FTP flows until 110.000 packets are successfully transmitted and split the simulation output in batches of size 10.000 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. For almost all data points, the width of the confidence intervals is below 5% of the measure's

[1] Note that this case rarely occurs, since *diff* takes positive real values whereas α is a natural number.

[2] As already noted in [12], the TCP Vegas implementation provided by ns-2 contains several subtle bugs, which we fixed.

value. As performance measure, we consider the goodput given by the number of bytes successfully transmitted divided by the length of each batch, the average number of packet retransmissions on the transport layer per flow, the average window size per flow, and the overall link layer dropping probability per flow. In the grid and random scenarios, we consider the measures aggregate goodput given by the sum of the goodput of individual flows as well as the individual goodput achieved by each flow.

As in [5], we consider Jain's fairness index given by:

$$\left(\sum_{i=1}^{n} x_i\right)^2 \Big/ n\sum_{i=1}^{n} x_i^2 \quad ,$$

where n is the number of flows and x_i denotes the goodput of the i-th flow.

Consistent with [5], in the simulation experiments conducted, all packet losses are due to link layer contention caused by hidden terminal effects. We do not observe buffer overflows in any performance experiment. Opposed to [5], our simulation study evidently shows that TCP Vegas with appropriately chosen parameters clearly outperforms TCP NewReno both for a short and a large number of hops.

4.2 Optimally Paced UDP over IEEE 802.11

In order to get intuition on the optimum achievable goodput over an IEEE 802.11 network for the chain topology and a given bandwidth, we consider an optimally paced UDP protocol as a transport agent. To define the packet transmission rate, we use a constant bit rate (CBR) traffic generator while setting the UDP packet size to 1460 bytes, equal to the TCP packet size we use through all of our simulations. Subsequently, we denote this transport protocols as paced UDP. Paced UDP shed also some light on the impact of link layer contention of IEEE 802.11 to a transport protocol for the chain topology.

To determine the optimum packet transmission rate for which paced UDP achieves the best channel utilization, we determine the minimal link layer propagation delay for 4 hops in a *h*-hop chain topology with a single flow as shown in Figure 1 of Section 4.3. That is, we calculate the 4-hop propagation delay for the first packet assuming a zero queuing delay. In order to keep the hidden terminal effects minimal, node *i* may only transmit packet p_j if packet p_{j-1} has been already forwarded by node *i+3,* where *i = 1,2,..,h-4*. Table 2 shows the 4-hop propagation delay for different bandwidths. Subsequently, we take the 4-hop propagation delay as an initial value for the time *t* between two successive packet transmissions for determining the optimum packet transmission rate. In an off-line simulation experiment, we increase *t* gradually until we observe the maximum goodput. No TCP variant can achieve as much goodput as paced UDP for the following two reasons: (1) the entire traffic and MAC overhead caused by the ACK flow is neglected in paced UDP; (2) paced UDP transmits packets with the optimal rate for each hop number, while TCP is window-based and has to probe for the available bandwidth. Furthermore, in paced UDP, we neglect packet retransmissions and determine the actual number of packets received by the UDP sink in terms of goodput.

Table 2: 4-hop propagation delay for different bandwidths

2 Mbit/s	5 Mbit/s	11 Mbit/s
29 ms	12 ms	8 ms

4.3 TCP Performance for h-hop Chain with a Single Flow

We consider an equally spaced chain comprising of *h*+1 nodes (*h* hops) with a single flow. Each node is 200 meters apart from each of its adjacent nodes. TCP packets travel along the chain from the leftmost node (i.e., the sender) to the rightmost node (i.e., the receiver). Figure 1 shows the *h*-hop chain topology with a single FTP flow without cross traffic. As observed in [5], successive packet transmissions of the single flow interfere with each other as they move along the chain. In fact, a potential sending node *i* constitutes a hidden terminal to an ongoing transmission from node *(i-3)* to *(i-2)* where *i = 4,5,..,h+1*. Node *i* cannot sense the ongoing transmission from *(i-3)* to *(i-2)* and thus starts transmitting, causing collisions with the ongoing transmission. Such hidden terminal effects result from the fact that the interference range of each node is much larger than its transmission range, and since the IEEE 802.11 protocol cannot achieve global packet scheduling between all nodes, such effects are inevitable with the standard IEEE 802.11 specifications. In the first experiment, we consider TCP Vegas with different values of the parameter α. The goal of this study lies in determining an optimal value of α for TCP Vegas without ACK thinning. Figures 2 to 4 show performance curves for TCP Vegas with $\alpha = 2, 3, 4$. In Figure 2, we observe that TCP Vegas with $\alpha = 2$ achieves the highest goodput for a chain length between 4 and 20 hops. For longer chains, the goodput of TCP is almost equal for all α values. Figure 3 shows that the average TCP window size increases for increasing α. Thus, TCP Vegas with $\alpha = 2$ has the smallest average window size. Figure 4 plots the goodput in a 7-hop chain for different bandwidths. We observe that for 2 Mbit/s bandwidth TCP Vegas with $\alpha = 2$ achieves the highest goodput. For 5.5 Mbit/s bandwidth TCP Vegas with $\alpha = 2$ yields only slightly higher goodput than the two other variants whereas for 11 Mbit/s bandwidth, all three TCP variants yield an equal goodput. Note that we observe a sub-linear growth of goodput with increasing bandwidth. This is because according to the IEEE 802.11 specifications, RTS, CTS and ACK control packets are sent at 1 Mbit/s regardless of the bandwidth used for data packets to achieve compatibility between different IEEE 802.11 versions. Thus, the relative overhead for sending control packets on the MAC layer increases with increasing data rate.

Fig. 1: 7-hop chain topology with a single flow

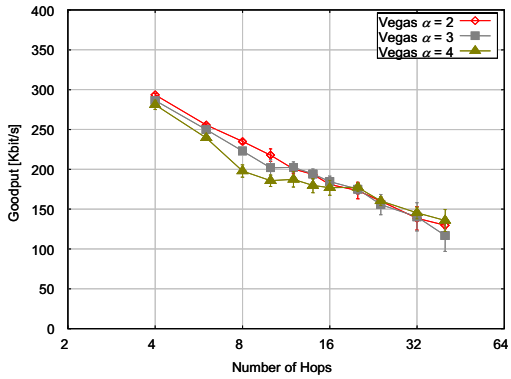

Fig 2: *h*-hop chain with 2 Mbit/s: TCP Vegas goodput vs. number of hops

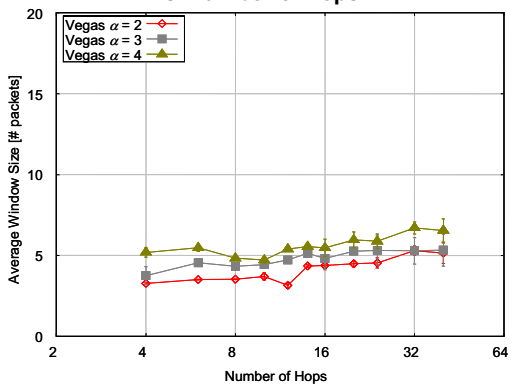

Fig 3: h-hop chain with 2 Mbit/s: TCP Vegas average window size vs. number of hops

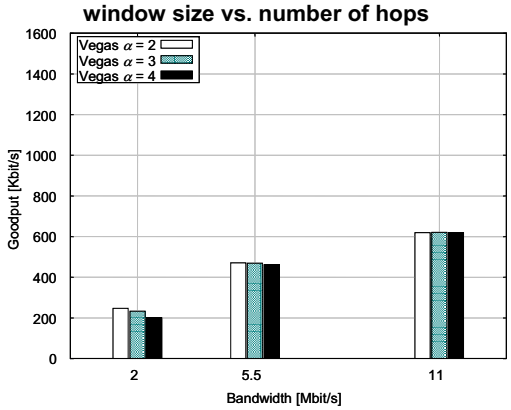

Fig 4: 7-hop chain: TCP Vegas goodput for different bandwidths

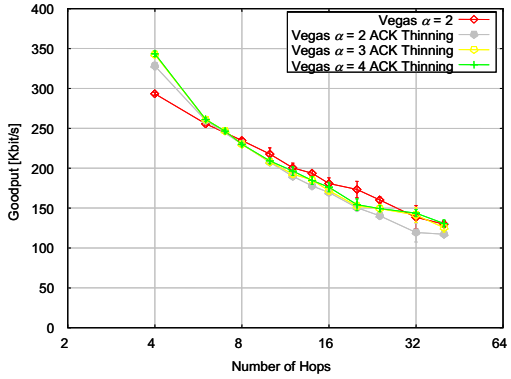

Fig 5: *h*-hop chain with 2 Mbit/s: TCP Vegas with ACK thinning: Goodput vs. number of hops

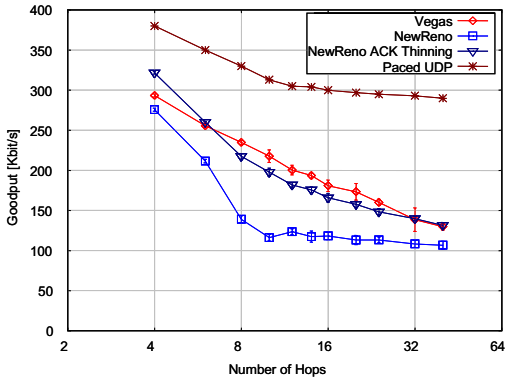

Fig 6: *h*-hop chain with 2 Mbit/s: Goodput vs. number of hops

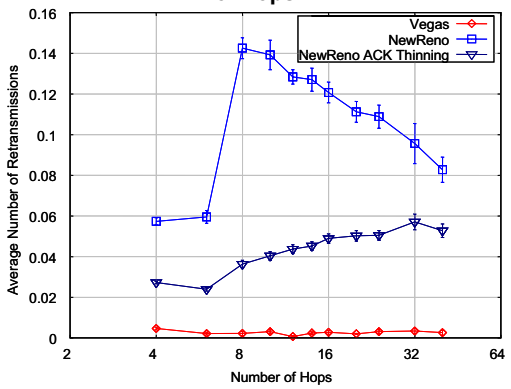

Fig 7: *h*-hop chain with 2 Mbit/s: Retransmissions vs. number of hops

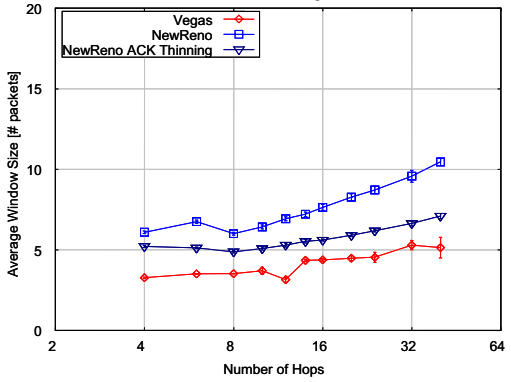

Fig 8: *h*-hop chain with 2 Mbit/s: Window size vs. number of hops

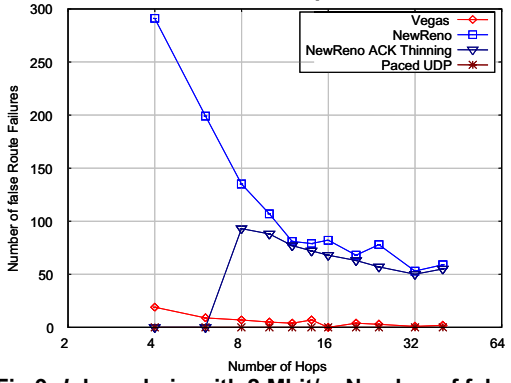

Fig 9: *h*-hop chain with 2 Mbit/s: Number of false route failures vs. number of hops

In the second experiment, we consider TCP Vegas with ACK thinning for different values of the parameter α. Again, the goal of this study lies in determining the optimal value of the parameter α. Figure 5 shows the goodput of TCP Vegas without ACK thinning with $\alpha = 2$ as well as for TCP Vegas with ACK thinning for different values of α. We observe that except for $h = 4$, TCP Vegas with $\alpha = 2$ performs slightly better than all other variants along all hops. Specifically, we notice that TCP Vegas with $\alpha = 2$ and ACK thinning performs slightly worse than TCP Vegas with $\alpha = 2$ for $h > 6$.

The reason for this performance difference is that the TCP window for TCP Vegas with $\alpha = 2$ and ACK thinning often decreases to 3, leading to a lack of acknowledgments at the TCP receiver which only acknowledges every fourth TCP packet for all packets with a sequence number greater than 8, as described in Section 3.2.

From the previous two experiments, we conclude that TCP Vegas with $\alpha = 2$ performs best for most number of hops and a bandwidth of 2 Mbit/s. Although increasing the bandwidth improves the performance of TCP Vegas with larger values of α due to the decreased contention on the MAC layer, TCP Vegas with $\alpha = 2$ remains the best choice.

In the third experiment, we consider TCP NewReno, TCP Vegas, TCP NewReno with ACK thinning, and paced UDP for the h-hop chain with varying hop count. As measures, we consider goodput, average number of retransmissions, average window size of the flow and number of false route failures as a function of chain length. The bandwidth is kept fixed to 2 Mbit/s. Figures 6 to 9 plot performance curves derived from this experiment. In Figure 6, we observe that TCP Vegas has up to 83% higher goodput than TCP NewReno (i.e., about 75% for 8 hops). Furthermore, for most number of hops, even TCP NewReno with ACK thinning performs slightly worse than TCP Vegas without ACK thinning. For both TCP Vegas and TCP NewReno with ACK thinning, the goodput decreases much slower with increasing number of hops than for TCP NewReno, indicating that both TCP Vegas and TCP NewReno with ACK thinning are significantly less sensitive to hidden terminal effects. From Figure 6, we also conclude that the goodput of TCP Vegas lies between 23% for 4 hops and 52% for 32 hops below the optimal goodput achieved by paced UDP, whereas the goodput of TCP NewReno lies between 28% for 4 hops and 63% for 32 hops below the goodput of paced UDP. Such big gap between both TCP variants and paced UDP outlines the significant impact of link layer interactions on the performance of TCP.

Figure 7 shows that TCP Vegas causes up to 99% less retransmissions than TCP NewReno. In fact, the number of retransmissions stays very low for TCP Vegas for any number of hops. Note that a reduction of retransmitted packets directly translates in a reduction of power consumption, which is a critical factor for resource constrained mobile devices. Opposed to that, the average number of retransmissions of TCP NewReno almost doubles from 6 to 8 hops reaching its peak and, subsequently, decreases gradually. This results from the fact that in a chain of seven and more hops, two hidden terminals may simultaneously disrupt the transmission of a single node, as it is the case for node 4 in a 7-hop chain. In contrast, a chain of up to six hops can produce at most a single hidden terminal effect for a single node. For TCP NewReno with ACK thinning, the average number of retransmissions is considerably lower than without ACK thinning. This is because ACK thinning results for TCP NewReno a smaller average window size as observed in Figure 8. Recall that during the slow start phase, TCP NewReno increases the window size dependent on the receipt of acknowledgments, specifically by one packet for each received ACK. In our simulations we have noticed that for $h \geq 7$, TCP NewReno operates during more than 40% of the connection in slow start. Since ACK thinning reduces the number of ACKs, this results in a less aggressive growth in the window size for TCP NewReno, and, thus, a smaller average window size. Figure 8 also shows that the average window size of TCP Vegas lies in the range 3.5 to 5.5 for increasing number of hops between 4 and 40, providing an explanation for the low number of retransmissions investigated in Figure 7. Comparing the average window size for longer chains to the optimum of $h/4$ [5], we find that TCP Vegas is close to the optimum for 32 hops, while it keeps the window size too small for longer chains. Recall that the parameter α determines the window size.

In order to get further insight in the impact of routing on the acquired results, we investigate the influence of false route failures on the performance of the examined TCP variants. False route failures result in case the link layer fails to deliver a packet to the next hop, either after seven unsuccessful transmissions for RTS control packets or after four unsuccessful transmissions for data packets. After the link layer notifies the routing layer about the transmission failure, the routing layer assumes that the route to the next hop is broken and thus deletes it from its routing table before broadcasting a route error message. In most such cases, the TCP sender times out and tries to retransmit the lost packet, initiating a new route discovery procedure, which causes additional traffic overhead. Figure 9 shows the number of false route failures for varying hop number. Consistent with the previous results, we observe that TCP NewReno causes significantly more false route failures than TCP Vegas, specifically 93% to 100%. That indicates that the larger average window size of TCP NewReno results in more packet drops on the link layer and thus more false route failures. For TCP NewReno with ACK thinning, we notice that it causes no false route failures for $h < 8$, then the curve increases sharply at $h = 8$. This effect is similar to what we have observed in Figure 7 for TCP NewReno at $h = 8$ and is due to the same reason that we have already mentioned at that point. Generally, all TCP variants, except for TCP NewReno with ACK Thinning at $h < 8$, experience less false route failures with increasing hop number. This is because the link layer contention decreases with increasing hop number since

packets in flight distribute more evenly among the nodes [5], leading to less packet drops and thus, to less false route failures. We conclude from Figures 6 to 9 that both TCP NewReno with ACK thinning and TCP Vegas are protocols of choice for improving TCP goodput in multihop chains and a bandwidth of 2 Mbit/s. Furthermore, in environments with limited power resources, TCP Vegas gains advantages over TCP NewReno with ACK thinning, since it reduces power consumption by avoiding unnecessary packet retransmissions and false route failures by using a smaller average TCP window size.

In the fourth experiment, we consider TCP NewReno, TCP Vegas, TCP NewReno with ACK thinning, TCP Vegas with ACK thinning, and paced UDP for a chain with 7 hops. We consider bandwidths of 2, 5.5, and 11 Mbit/s. Furthermore, we consider TCP NewReno, for which we bound the TCP window size artificially as proposed in [5]. The maximum window allowed, MaxWin, is optimized for a chain topology with 7 hops. Consistent with [5], we found MaxWin = 3 for all bandwidths, i.e., with MaxWin = 3, TCP NewReno reaches the highest goodput for $h = 7$. Again, goodput increases sub-linearly with increasing bandwidth. To determine the optimal transmission rate for paced UDP, Figure 10 shows the goodput of paced UDP for different times t between two successive packet transmissions. The experiments show that paced UDP achieves optimal goodput for t_{opt} = 35.7ms. Consistent with [9], we find that goodput drops rapidly when t gets smaller than t_{opt}, while it degrades graciously when t exceeds t_{opt}. That is, for $t < t_{opt}$ the transmission rate is too high causing increased link layer contention due to hidden terminal effects. For $t > t_{opt}$, link layer contention is minimal, but the rate decreases linearly causing such gracious goodput decrease. We conclude from Figure 10 that the optimal pacing rate is extremely sensitive to network conditions. Thus, each effective rate-based transport protocol will require a low-latency algorithm for quickly adapting the transmission rate to changing network conditions.

As primary performance measures, we consider the goodput, which is shown for different bandwidths in Figure 11. To get deeper insight in how goodput is achieved by the individual TCP variants, we furthermore investigate the average number of retransmissions, the average window size of the flow as well as the overall link layer dropping probability (averaged over all intermediate node), which are shown in Figures 12 to 14. Recall that the bars for 2 Mbit/s exactly represent the results for the 7-hop chain in Figures 6 to 8. Extending the findings of Figures 6 to 8, we find that applying ACK thinning in TCP Vegas does not improve goodput at 2 Mbit/s, but reduces drops on the link layer, as shown in Figure 14. However, reduction of link layer drops does not translate in increased goodput, since link layer drops are not visible to TCP Vegas on the transport layer, as shown in Figure 12. This indicates that the load on the link layer is moderate, so that all packets can be sent after a few retries. In fact, applying ACK thinning in Vegas will even result in an increased number of packet retransmissions on the transport layer, since a missing ACK may result in the retransmission of multiple packets. With increasing network bandwidth, packet

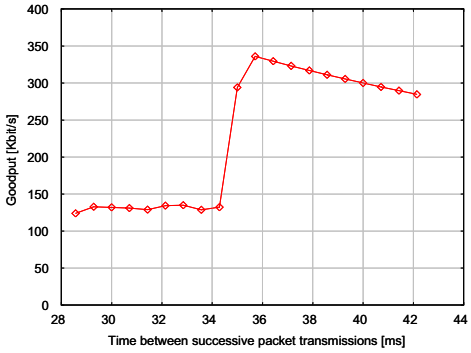

Fig 10: 7-hop-chain with 2 Mbit/s: Goodput vs. packet inter-sending time

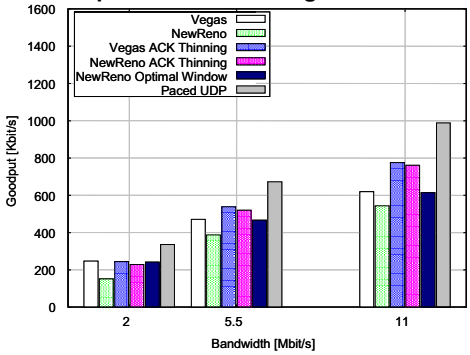

Fig 11: 7-hop chain: Goodput for different bandwidths

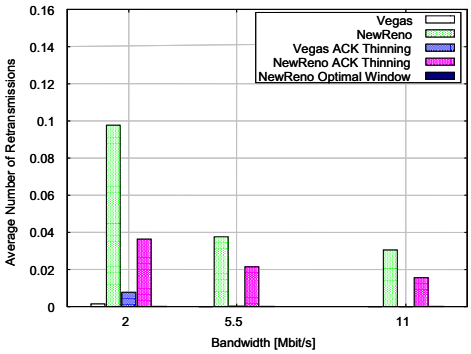

Fig 12: 7-hop chain: Retransmissions for different bandwidths

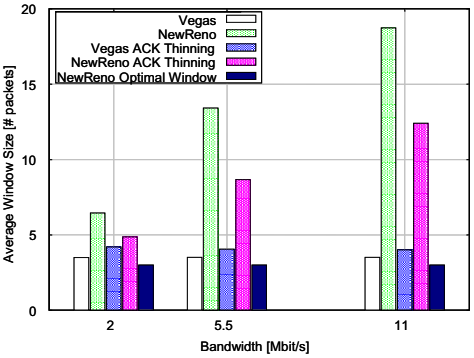

Fig 13: 7-hop chain: Window size for different bandwidths

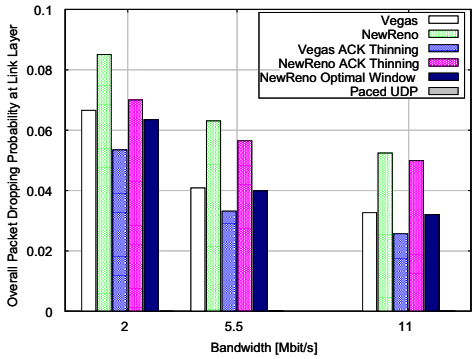

Fig 14: 7-hop chain: Packet dropping probability for different bandwidths

retransmissions on the transport and packet drop probability on the link layer decrease, since data packets can be transmitted in shorter time, reducing the probability for packet collisions. Furthermore, the improvement achieved by ACK thinning increases for both TCP NewReno and TCP Vegas, since fewer collisions of data packets with ACK packets enable better utilization of available bandwidth. Comparing TCP Vegas to the other TCP variants, we find that it performs significantly better than TCP NewReno, and as good as TCP NewReno with optimal window for all bandwidths. Nevertheless, TCP Vegas is outperformed by both TCP Vegas with ACK thinning and TCP NewReno with ACK thinning for increasing bandwidth availability, with a gap of about 20% in goodput for 11 Mbit/s. Comparing both TCP Vegas with ACK thinning and TCP NewReno with ACK thinning to the optimal goodput achieved by paced UDP, we find that both are close to the optimum with a goodput gap of at most 32% at 2 Mbit/s, and only 23% at 11 Mbit/s. Considering power consumption, we find that both TCP Vegas and TCP Vegas with ACK thinning are superior to the TCP NewReno variants, since they reduce packet retransmissions on the transport layer. Furthermore, TCP Vegas with ACK thinning has the least link layer drops among all variants. We conclude from Figures 11 to 14 that both TCP Vegas with ACK thinning and TCP NewReno with ACK thinning are the protocols of choice for improving TCP goodput in a chain scenario. In environments with limited energy, TCP Vegas with ACK thinning gains advantages over TCP NewReno with ACK thinning, since it reduces power consumption by avoiding unnecessary packet retransmissions on the one hand and conserves the shared radio resources by using a smaller TCP window size on the other hand.

4.4 TCP Performance in more complex Topologies with several concurrent Flows

In this section, we evaluate the examined TCP variants in more complex scenarios with multiple concurrent flows.

4.4.1 Grid Topology

Figure 15 shows the node distribution and flow patterns used for the grid simulation. The grid consists of 21 nodes, whereas all horizontally and vertically adjacent nodes are 200 meters apart. We consider a total of six competing FTP flows, three horizontal and three vertical. In such topology, all flows interfere with each other, increasing contention on the link layer. In this simulation, we do not only consider the aggregate goodput over all flows, but also the achieved goodput of each flow as well as the fairness degree for each of the examined TCP variants. Figure 16 plots the aggregate goodput of TCP Vegas and TCP NewReno for different bandwidths, both with and without ACK thinning. We observe that for 2 Mbit/s, TCP NewReno slightly outperforms TCP Vegas, whereas for 5.5 and 11 Mbit/s, both variants have almost equal aggregate goodput. Figure 16 further shows that applying ACK thinning for TCP Vegas does not yield any performance improvement for 2 Mbit/s, which is consistent with the results of the chain simulation. However, as bandwidth availability increases, the performance of TCP Vegas with ACK thinning improves over the performance of TCP Vegas. As for TCP NewReno with ACK thinning, its goodput also increases with increasing bandwidth, achieving higher values. However, as we are regarding a topology with multiple flows, the fairness factor plays a significant role in specifying the performance of a TCP variant. Due to the absence of global scheduling of IEEE 802.11, there exists a trade-off between the fairness between TCP flows and the aggregate goodput over all flows. That is, the more fairness is achieved, the more suffers the aggregate goodput, since the available bandwidth is not optimally used due to the increased contention between the TCP flows. Similar observations regarding such trade-off between fairness and aggregate goodput were made in [14]. In order to investigate the fairness of the examined TCP variants, we consider the goodput of each flow for a fixed bandwidth of 11 Mbit/s. Observing the results in Figure 17, we see that while TCP Vegas and TCP NewReno achieve almost an identical aggregate goodput, the flows of TCP Vegas achieve more fairness than the flows of TCP NewReno. Using TCP NewReno, flows

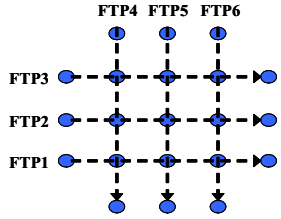

Fig 15: 21-node grid topology with 6 competing flows

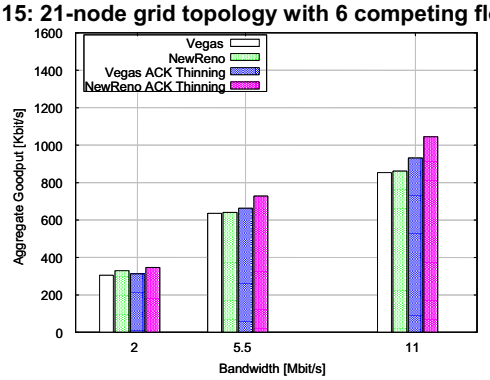

Fig 16: Grid topology: Aggregate goodput for different bandwiths

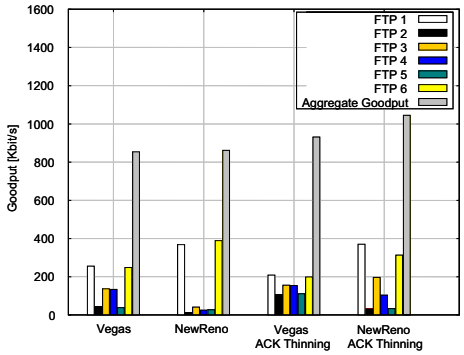

Fig 17: Grid topology: Single goodput of each flow and aggregate goodput over all flows for different TCP variants at 11 Mbit/s

Table 3: Grid topology: Jain's fairness index

	TCP Vegas	TCP NewReno	TCP Vegas w/ ACK Thinning	TCP NewReno w/ ACK Thinning
2 Mbit/s	0.54 [0.53 : 0.55]	0.32 [0.31 : 0.33]	0.69 [0.68 : 0.70]	0.40 [0.38 : 0.42]
5.5 Mbit/s	0.64 [0.60 : 0.68]	0.43 [0.40 : 0.46]	0.87 [0.84 : 0.90]	0.56 [0.52 : 0.60]
11 Mbit/s	0.73 [0.69 : 0.77]	0.52 [0.48 : 0.56]	0.94 [0.90 : 0.98]	0.63 [0.60 : 0.66]

one and six achieve the highest goodput on cost of the remaining flows, which basically starve. However, using TCP Vegas, flows one and six achieve less goodput, sacrificing more bandwidth for the remaining flows, and thus achieving more fairness. Regarding TCP NewReno with ACK thinning and TCP Vegas with ACK thinning, we observe similar effects. Although TCP NewReno with ACK thinning outperforms all other variants in terms of aggregate goodput, it achieves less fairness than TCP Vegas with ACK thinning. In fact, TCP Vegas with ACK thinning achieves the best fairness of all variants with only 10% less aggregate goodput than TCP NewReno with ACK thinning. Previous studies [14] have reported up to 42% less aggregate goodput in order to achieve near-optimal fairness. To formally investigate the fairness among all six TCP flows, we calculate Jain's fairness index for each variant and bandwidth as defined in Section 4.1. The results are shown in Table 3. Recall that a fairness index of $1/n$ indicates worst fairness among n flows, while a fairness index of 1 indicates optimal fairness. The values in the table confirm our previous findings. Furthermore, we notice that not only ACK thinning improves the fairness of TCP Vegas and TCP NewReno, but also increasing the bandwidth, since both reducing the TCP ACKs as well as increasing the bandwidth result in less contention on the link layer, and thus less competition between the flows.

Consistent with the results of Section 4.3, we conclude from Figures 16 and 17 as well as from Table 3 that TCP Vegas with ACK thinning is the protocol of choice for achieving the best trade-off between aggregate goodput and fairness. Among all examined variants, TCP Vegas with ACK thinning achieves best fairness results and has only 10% less aggregate goodput than TCP NewReno with ACK thinning.

4.4.2 Random Topology

As a third topology, we consider a random topology of 120 nodes uniformly distributed on an area A = 2500x1000 m^2. We set 10 FTP connections that run simultaneously, with all FTP sources and destinations randomly selected. According to [2], all nodes in the network can communicate with each other over one or more hops with probability P = 99.9%.

Figure 18 plots the aggregate goodput for the TCP variants at different bandwidths. Consistent with the results for the previous topology, we see that TCP Vegas and TCP NewReno achieve similar goodput for all bandwidths, with a maximum of 3% more goodput for TCP Vegas at 11 Mbit/s. Applying ACK thinning for both TCP Vegas and TCP NewReno also improves the goodput for increasing bandwidth. Different from the cases for 2 Mbit/s and 5.5 Mbit/s where TCP NewReno with ACK thinning slightly outperforms TCP Vegas with ACK thinning, both variants achieve identical goodput for 11 Mbit/s. Figure 19 shows the goodput of each flow as well as the aggregate goodput over all flows for all examined TCP variants and a bandwidth of 11 Mbit/s. Compliant with the results for the grid topology, we observe that by using TCP NewReno, the fourth flow gets the highest fraction of the available bandwidth on cost of the other flows, letting flows three and eight completely starve. Overall, TCP Vegas achieves more fairness than TCP NewReno, and applying ACK thinning further improves fairness, letting TCP Vegas with ACK thinning achieve the best fairness among all variants. Extending our findings of Figure 17, we observe that

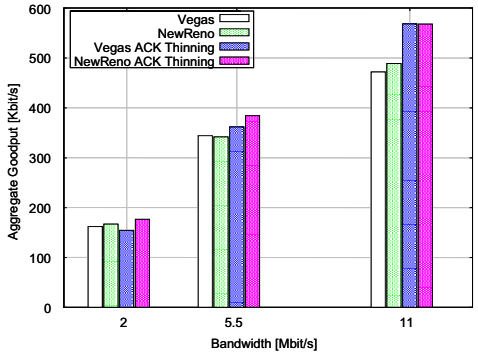

Fig 18: Random topology: Aggregate goodput for different bandwiths

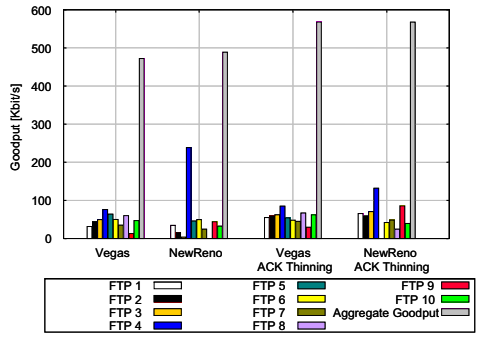

Fig 19: Random topology: Single goodput of each flow and aggregate goodput over all flows for different TCP variants at 11 Mbit/s

Table 4: Random topology: Jain's fairness index

	TCP Vegas	TCP NewReno	TCP Vegas w/ ACK Thinning	TCP NewReno w/ ACK Thinning
2 Mbit/s	0.43 [0.41 : 0.45]	0.22 [0.20 : 0.24]	0.62 [0.60 : 0.64]	0.40 [0.38 : 0.42]
5.5 Mbit/s	0.80 [0.77 : 0.83]	0.68 [0.64 : 0.72]	0.87 [0.85 : 0.89]	0.70 [0.67 : 0.73]
11 Mbit/s	0.87 [0.84 : 0.90]	0.72 [0.70 : 0.74]	0.90 [0.88 : 0.92]	0.74 [0.71 : 0.77]

TCP Vegas with ACK thinning achieves the same aggregate goodput as TCP NewReno with ACK thinning in spite of its best fairness results. In this case, TCP Vegas with ACK thinning achieves the best trade-off between aggregate goodput and fairness. Equivalent with the results for the grid topology, Table 4 confirms that both applying ACK thinning and increasing the bandwidth availability increases fairness for all variants. This simulation confirms that TCP Vegas with ACK thinning is the protocol of choice for all bandwidths and multi-flow environments. Figure 19 further extends our findings by showing that TCP Vegas with ACK Thinning can achieve best fairness results without sacrificing aggregate goodput in comparison to TCP NewReno with ACK thinning. From Table 3 and Table 4 we conclude that TCP Vegas achieves between 21% and 95% more fairness than TCP NewReno, whereas TCP Vegas with ACK thinning yields a fairness improvement of 22% to 73% compared to TCP NewReno with ACK thinning.

Conclusions

We showed that in static multihop wireless networks using IEEE 802.11, TCP Vegas with parameters $\alpha = \beta = 2$ clearly outperforms TCP NewReno, which is widely deployed in wired networks. In fact, TCP Vegas achieves up to 83% higher goodput and up to 99% less packet retransmissions. Consistent with [1], we find that ACK thinning substantially increases the performance for TCP NewReno for persistent flows over IEEE 802.11 with 2 Mbit/s bandwidth. However, we observe that this improvement is not due to the reduced number of link-layer packet collisions triggered by hidden terminal effects. In fact, the key driver why ACK thinning improves the performance of TCP NewReno constitutes the fact that ACK thinning considerably reduces the average window size and, thus, helps TCP NewReno stay closer to the optimum window size for multihop wireless networks. TCP Vegas with appropriately chosen parameters already keeps its window size close to the optimum. As a consequence, ACK thinning yields almost no goodput improvement for TCP Vegas over IEEE 802.11 with 2 Mbit/s bandwidth. For TCP Vegas over IEEE 802.11 with 5.5 Mbit/s and 11 Mbit/s bandwidths, ACK thinning yields up to 25% goodput improvement, because the reduced ACK stream causes better channel utilization for data packets.

We find that TCP Vegas achieves better fairness than TCP NewReno and that both applying ACK thinning and increasing the bandwidth availability yield further fairness improvement, letting TCP Vegas with ACK thinning achieve the best fairness results among all examined variants for all bandwidths, with 24% to 73% more fairness than TCP NewReno with ACK thinning.

Finally, it is noteworthy that the substantially reduced amount of packet retransmissions of TCP Vegas and TCP Vegas with ACK thinning results in significant savings of energy consumption. Thus, if we consider both goodput and fairness, the transport protocol of choice for ad hoc networked PDAs and other mobile devices with restricted energy resources should be TCP Vegas with ACK thinning in case of 2, 5.5 and 11 Mbit/s bandwidths, respectively.

References

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

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

[3] L.S. Brakmo and L.L. Peterson, TPC Vegas: End-to-End Congestion Avoidance on a Global Internet, *IEEE Journal on Selected Areas in Comm.*, **13**, 1995.

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

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

[6] G. Hasegawa, M. Murata, and H. Miyahara, Fairness and Stability of Congestion Control Mechanisms, *Proc. GLOBECOM*, Rio de Janeiro, Brazil, 1999.

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

[8] IEEE Standard 802.11g, available at http://standards.ieee.org/getieee802/802.11.html.

[9] J. Li, C. Blake, D.S. De Couto, H.I. Lee, and R. Morris, Capacity of Ad Hoc Wireless Networks, *Proc. ACM MOBICOM*, Rome, Italy, 2001.

[10] S.H. Low, L.L. Peterson, and L. Wang, Understanding TCP Vegas: A Duality Model, *Proc. ACM SIGMETRICS/Performance*, Cambridge MA, 2001.

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

[12] C. Samios and M. Vernon, Modeling the Throughput of TCP Vegas, *Proc. ACM SIGMETRICS*, San Diego CA, 2003.

[13] F. Wang and Y. Zhang, Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response, *Proc. ACM MOBIHOC*, Lausanne, Switzerland, 2002.

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

[15] S. Xu and T. Saadawi, Performance evaluation of TCP algorithms in multi-hop wireless packet networks, *Wireless Communications and Mobile Computing*, pages 85 – 100, 2002.