2012-03-05

Feasibility of TCP for Wireless Mesh Networks

Richard Lloyd Lee
Brigham Young University - Provo

Feasibility of TCP for Wireless Mesh Networks

Richard Lloyd Lee

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Daniel Zappala, Chair
Kent Seamons
Christophe Giraud-Carrier

Department of Computer Science

Brigham Young University

April 2012

ABSTRACT

Feasibility of TCP for Wireless Mesh Networks

Richard Lloyd Lee
Department of Computer Science, BYU
Master of Science

When used in a wireless mesh network, TCP has shortcomings in the areas of throughput and fairness among traffic flows. Several methods have been proposed to deal with TCP's weakness in a wireless mesh, but most have been evaluated with simulations rather than experimentally. We evaluate several major enhancements to TCP – pacing, conservative windows, and delayed ACKs – to determine whether they improve performance or fairness in a mesh network operating in the BYU Computer Science building. We also draw conclusions about the effectiveness of wireless network simulators based on the accuracy of reported simulation results.

Keywords: Wireless mesh, transport protocols, experimental evaluation, TCP, simulations

ACKNOWLEDGMENTS

The author would like to thank his wife, Kirsti and their daughter, Maggie, for their support and patience during the completion of this thesis. Thanks are also due to the entire graduate committee for their invaluable feedback, as well as the BYU Computer Science staff for their Herculean efforts in helping to complete the thesis process in a timely fashion. Finally, Randy Buck has the author's gratitude for his tireless work on the WiFu framework that was so critical for this thesis.

Table of Contents

Chapter 1

Introduction

Wireless mesh networks consist of many nodes that communicate wirelessly with one another; that is, there is no central access point or base station around which the network is centered. Packets are forwarded from node to node until they reach their destination, which may be another node in the network, or a gateway to the Internet. Due to their low cost, such networks are useful for providing Internet connections in areas in which installing infrastructure is prohibitively expensive; for example, urban areas where in-ground infrastructure is difficult to install, and developing countries, where the resources to build infrastructure simply are not available. They are also used to set up temporary, low-cost, and quickly available networks for emergency situations such as evacuation centers or disaster relief efforts, or for sporting events [8, 12, 2]. Mobile users are also benefitted by such networks, provided they remain within range of some other nodes within the mesh network. Wireless mesh networks have already been deployed in places such as Berlin [3], Cambridge [4], Champaign-Urbana [1], San Francisco [6], Seattle [5], and Southampton [7]. Bruno, Conti, and Gregori [12] note a number of companies based on mesh network technologies as well as commercial applications.

The link layer of wireless mesh networks presents difficult problems for TCP. TCP's congestion control algorithm operates under the assumption that dropped packets occur due to network congestion. In traditional wired networks, this is generally a correct assumption; however, in wireless networks, there are many sources of interference that can cause packets to be corrupted or lost. In multihop networks, the problem is compounded because multiple nodes in the network (or even on the same path) will contend with one another for the wireless medium [17]. Moreover, it has been shown that interaction between TCP and the 802.11 MAC can lead to severe unfairness [18, 11, 32, 20].

There is a body of work focused on improving the link-layer conditions. Parsa and Garcia-Luna-Aceves [28] propose TULIP, a link-layer protocol which uses timers based on maximum link propagation delay in lieu of round-trip-time calculations. TULIP does not require a base station and is proposed as suitable for wireless mesh networks, although their simulations did include a base station for the purposes of comparison with other well-known protocols that deal specifically with that situation. In their paper presenting the MACAW protocol, Gerla, Tang, and Bagrodia [19] propose that link-level reliability mechanisms such as link-level ACKs and less-aggressive MAC backoffs (instead of exponential backoffs) improve regular TCP's performance in wireless mesh networks.

Changing the transport layer, however, would be simpler and a more cost-effective solution than altering the link layer. Any new protocol, whether at the transport or link layer, would need to be widely adopted in order to make a difference; a link-layer protocol would need to be implemented in hardware that was then widely adopted, whereas modifying TCP requires only software changes. This means that existing hardware can be used to implement and deploy wireless mesh networks using modified TCP, allowing for fast adoption without the expense and delay of building, acquiring, and deploying new hardware.

A number of approaches have been proposed to significantly improve TCP's behavior in wireless mesh networks based on promising simulation results, but few have been implemented and tested experimentally in a mesh network. It is well-known that simulators use many simplifications when simulating wireless network environments which could call their accuracy into question [29, 21, 25]. Suggested improvements to TCP include the use of pacing to control TCP's send rate [15], imposing a more conservative window increment [27], and combining ACKs [9].

In this thesis, we evaluate pacing, conservative windows, and delayed ACKs using experiments on a wireless mesh network. We create implementations of each protocol using the WiFu test framework discussed in Section 3.2. and test their performance on the wireless mesh. In doing so, we determine whether TCP can be effectively adapted to wireless multihop networks and how this may be accomplished. We find that pacing has potential to improve fairness, but TCP-AP [15] is outperformed on goodput because of the over-conservative metric it uses. Conservative windows yield no improvement over TCP Tahoe; they are simply not a suitable approach for wireless mesh networks. Delayed ACKs do not improve fairness, but improve TCP goodput considerably. Our

findings indicate that a proper pacing metric combined with the goodput increase of delayed ACKs would be a considerable improvement on TCP, but further work is needed to determine what metric is appropriate and to verify that both goodput and fairness improvements occur. We also provide insight on the effectiveness of simulation for developing transport protocols for wireless networks, as most of the evaluated simulation-based assertions are found not to hold up in an actual wireless mesh network.

1.1 TCP Background

TCP Tahoe is the original incarnation of TCP featuring a congestion window that limits the number of outstanding packets and grows that number as packet acknowledgements are received. If a packet is dropped or delayed, that window is reduced to one maximum segment size (MSS), or in other words, a single packet. TCP NewReno is a newer standard which reduces the window only by half of the current size when three packets in a transmission are delayed. Dropped packets still cause the window to reset as in TCP Tahoe.

Chapter 2

Related Work

We investigate the claims of three papers that have reported major improvement to TCP.

Sundaresan, Anantharaman, Hsieh, and Sivakumar [30] posit that window-based transmission such as TCP's congestion control causes bursty traffic. They further contend that this burstiness leads to inaccurate RTT estimates for packets later in the burst, which leads to poor channel usage. ElRakabawy, Klemm, and Lindemann [15] propose an adaptive pacing scheme to improve wireless multihop throughput using an estimate of the four-hop propagation delay to determine an optimal sending rate. That is, they incorporate a rate limiter into TCP and adjust its rate using estimated values of the propagation delay to the node four hops away on the path to the destination. This estimate is provided by analyzing covariance on a window of observed round trip times. Using simulations in the ns-2 simulator, they find up to an 84% increase in goodput over TCP NewReno.

Elrakabawy and Lindemann [14] continue to implement adaptive pacing in a test network and propose a modification to their metric known as out-of-interference delay (OID). OID is defined as follows:

$$OID = (H_c s + 1)(t_q + \frac{s_{data}}{b}) \tag{2.1}$$

where $H_c s$ is the carrier sensing range in number of hops, t_q is the average queuing delay per node, s_{data} is the size of a packet, and b is the bandwidth. They claim up to a tenfold improvement in throughput over TCP-NewReno while maintaining fairness in their test network.

Nahm, Helmy, and Kuo [27] find that routing dynamics and, by extension, TCP performance are adversely affected by TCP's aggressive behavior. They propose a fractional window increment called FeW as a solution. The fractional increment would make TCP more conservative in sending packets; there would be fewer instances of network congestion, fewer link-layer contention issues, and fewer cases of overreaction by routing protocols, all of which harm TCP performance. In FeW, the

sending window W is incremented upon the receipt of an ACK according to the following equation:

$$W_{new} = W_{current} + \frac{\alpha}{W_{current}} \tag{2.2}$$

where $0 \leq \alpha \leq 1$. It can be seen that setting $\alpha = 1$ results in the normal TCP behavior of incrementing the sending window by 1 every time $W_{current}$ ACKs are received. They validate their approach in a few ns-2 simulations with $\alpha = 0.01$, where they find that FeW has a more than 90% increase in throughput over a normal TCP congestion control mechanism.

Another observation that Sundaresan et al.[30] make is that TCP's use of ACKs for congestion control and reliability leads to a large amount of control traffic overhead, which will contend with data traffic if the forward and reverse paths in the wireless mesh are the same. If they are not the same, however, there is still the problem of TCP relying on ACKs and thus relying on both the forward and reverse paths. These paths may even continue to contend with each other when the paths contain nodes that are physically close to nodes in another path. Altman and Jiménez [9] also observe that TCP ACKs contribute to a high amount of transmission overhead in a network using RTS and CTS. They therefore propose generating a single ACK after two to four TCP packets have been received, or after a timeout interval has expired. This approach was simulated using ns-2 and found to improve throughput by 20 to 40%. They build on that result by dynamically changing the number of packets to include in a single ACK, using frequent ACKs early in a TCP session to build up the sending rate, and fewer later, when TCP would be more prone to overcorrection of packet loss.

A separate body of work has focused on mobility in wireless mesh networks. Such work must deal with a higher frequency of actual route failure and recalculation of routes rather than loss due primarily to interference or contention. ATP [30], ELFN [22], and ATCP [26] all attempt to deal with this phenomenon, as do some TCP variants such as DOOR [31]. There are also link-layer enhancements like EPLN/BEAD [33] which attempt to use caching with a distributed cache updating strategy and cross-layer information to improve performance in a mobile wireless mesh.

Karsten Reineck's evaluation of simulation tools [29] demonstrates the need for experiments in addition to simulations in wireless scenarios. He finds that interference issues are fairly well-

modeled, but all evaluated simulators (including ns-2) do a poor job of simulating gradual signal degradation over distances; the signal is assumed to be equally viable at any distance considered "in range" of a transmitter. Knowing which details of our wireless model are important to simulate is a difficult but important problem; Heidemann, Bulusu, and Elson [21] address this problem and recommend some techniques to assist in finding and understanding the proper level of detail for a simulation. Kotz et al. [25] present a list of problematic assumptions that wireless network simulators make:

1. Assume a flat, level world.
2. All radios have perfectly circular transmission areas.
3. All radios have the same effective range.
4. If a radio can sense another radio, that other radio can sense the first.
5. There is no partial sensing; if a radio can be heard at all, then it is heard perfectly.
6. Signal strength is determined only by distance.

Each of these assumptions is problematic in the real world, and Kotz et al. demonstrate that they have adverse effects on the quality of simulations operating under them. Since these results all cast some level of doubt on results obtained via simulations, work to experimentally validate the results of simulations in real mesh networks is vital.

Chapter 3

Protocol Descriptions

3.1 Introduction

In our work, we determine whether TCP can be modified for improved performance in wireless mesh networks. The WiFu transport protocol development framework will be used to experimentally evaluate three approaches for improving TCP in wireless mesh networks. Pacing involves limiting the transmission rate of a TCP sender in order to reduce burstiness, resulting in more efficient wireless channel usage. Conservative windows involve reducing the rate at which the sender's congestion window, cwnd, grows; this approach aims to thus reduce the amount of traffic contending for the wireless medium. Delayed ACKs attempt to reduce the amount of TCP overhead by combining a number of acknowledgments into one bulk acknowledgement.

3.2 WiFu Framework

WiFu is a new transport protocol framework developed in BYU's Internet Research Lab [13]. It allows transport protocols to be easily and modularly written and run in user-space, as opposed to the kernel. One of the key features of WiFu is that transport protocols can be divided into a set of modules; TCP, for example, can be composed of the connection manager, reliability, and congestion control modules. This enables both faster development and simpler debugging of transport protocols. Using this framework will make it simple to both extend TCP and combine various modules to determine how the combination of TCP variants behave.

3.3 TCP Tahoe

We opt to use the TCP Tahoe protocol already implemented in WiFu as a control to test pacing, conservative windows, and delayed ACKs. The difference between Tahoe and the NewReno protocol

that [15], [27], and [9] compare to is a small modification of the congestion control behavior [16]. Since the idea behind each of these proposed approaches for improving TCP in a wireless mesh is to avoid triggering the congestion control portion of TCP unnecessarily, an improvement in throughput in Tahoe translates to an improvement in throughput in NewReno. Fairness is likewise preserved because both versions of TCP are additive-increase, multiplicative decrease (AIMD).

3.4 Pacing

The primary additions to TCP for our pacing experiments are a rate limiter and a delay estimator. The rate limiter takes effect just before a packet is sent into the network; it is a simple 1-capacity token bucket guaranteeing a maximum send rate. That is, a packet will be sent, but no further packet may be sent until a specified time delay has passed. Further incoming packets will be queued and transmitted at the specified rate in the order in which they were received at the limiter. The delay estimator calculates a rate to adjust the rate limiter to upon the receipt of an ACK based on the four-hop-delay (FHD) metric from [15]. The delay estimator may also be overridden to use a fixed delay in lieu of the delay calculation in order to make the rate limiter send at a constant rate.

We test the claims of TCP-AP [15] noted in Section 2, which indicate that we should observe dramatically increased throughput in our network implementation of this protocol. We examine various pacing rates to determine whether the four-hop delay estimate is the best heuristic for pacing. The modular nature of the WiFu framework allows us to easily create several rate estimators/calculators using different n-hop heuristics in order to determine if the four-hop delay is actually the best estimate to use. The idea behind the four-hop delay is that it is a distance at which the initial node should no longer have to contend for the wireless medium with other nodes along the sending route, so we test several values of n in an effort to find the correct average distance of non-interference for our mesh network. A number of fixed rates are also implemented so that we may determine whether pacing is a useful approach independent of the n-hop delay metric.

3.5 Conservative Windows

TCP-Tahoe's congestion control module is modified to allow alternate window increments as described in TCP-FeW [27]. The new congestion control module increments the sending window

size by only fractions, as seen in our discussion of FeW [27] in Section 2. If the window size w is smaller than 1, then TCP is rate-limited to send only one packet every $\frac{RTT}{w}$ seconds. Otherwise, the window is considered to be the maximum of either the integer floor of the fractional window size or a minimum window size parameter. A minimum window size is used as recommended in [27] to avoid a long idle period when the window size is small. In order to facilitate comparison with TCP-Tahoe, we choose the minimum window size to be 1 MSS. We use this congestion control module to verify the throughput claims in [27].

3.6 Delayed ACKs

The dynamic delayed ACK is implemented as a modification of the TCP Tahoe reliability module in WiFu. A series of thresholds are defined in the ACK logic that determine how many ACKs should be grouped together. The thresholds are based on the sequence numbers of incoming packets as in [9] and gradually increase the number of combined ACKs from 1 to 4. After the requisite number of data packets have been received or a timeout value is reached, the ACK packet is sent. We fix our timeout value at .1 seconds to match [9] and fix our threshold values $l1$, $l2$, and $l3$ at sequence numbers 1000, 10000, and 100000, respectively. Since we use a one-MB file transfer for our test, this means that 90% of each transfer is performed combining the maximum number of ACKs (4).

A bug was found where the transfer ramps up faster than we intended; the thresholds go straight from 1 to 3 ACKs, then continue at 3 ACKs for the next threshold. If anything, however, this strengthens our result; it seems that ramping up to a full delayed ACK behavior faster is, in fact, better for throughput. See discussion in 5.5.

3.7 Combinations

The WiFu framework used for these experiments makes combining approaches a simple matter of combining modules into new experimental protocols. We discuss combined behavior in Section 5.6.

Chapter 4

Validation Methodology

4.1 Topologies

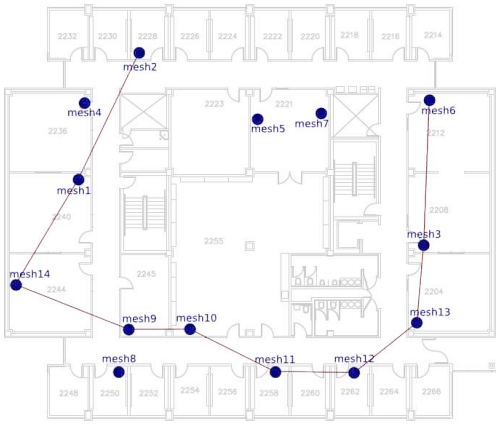
We use a number of topologies to evaluate protocol performance in the aspects of goodput and fairness. All topologies are run on two floors in the computer science building on a wireless mesh network using 802.11a on channel 36 with MAC-layer retries set to zero, RTS/CTS turned off, and static routing. The 802.11a range was chosen to minimize interference from devices around the building. Different topologies are constructed by altering the transmission power and rate on the individual nodes.

4.1.1 Chain Topologies

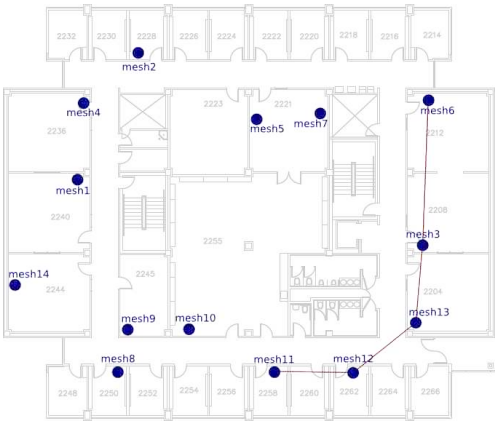
All single-chain experiments consisted of one, two, and five simultaneous flows along a chain. The transmission power for each network adapter is set to 6 dBm. Transmission rates were fixed at 24 Mbps. Nodes not involved in the chain were turned off or to orthogonal channels to minimize interference from sources outside the chain. The long chain topology consists of 9 hops along a path from mesh6 to mesh2 as seen in Fig. 4.1(a). The short chain topology consists of 4 hops from mesh6 to mesh11 (see Fig. 4.1(b)). The one-hop topology consists of a single-hop path from mesh6 to mesh3.

4.1.2 Parallel Chains

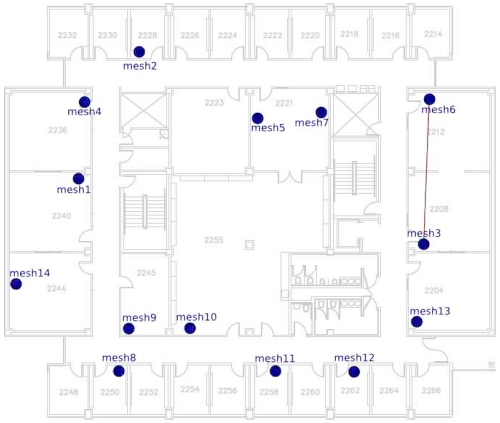
We create a parallel chain topology with four-hop flows along vertically parallel paths on the first and second floors as seen in Fig. 4.2. Transmission power for each chain is fixed at 10 dBm and the rate is set to 12 Mbps. The increased power gives us the property that although the two paths do not intersect, they can cause interference at parallel nodes.



(a) 9-hop chain

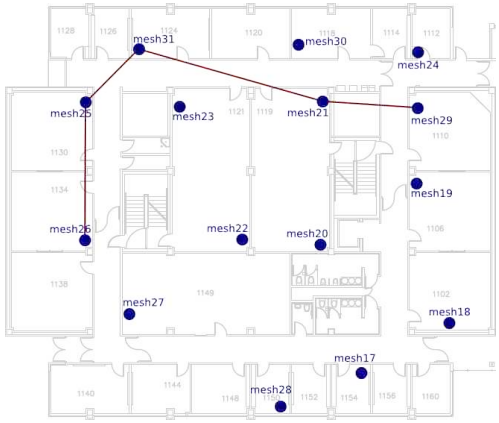


(b) 4-hop chain

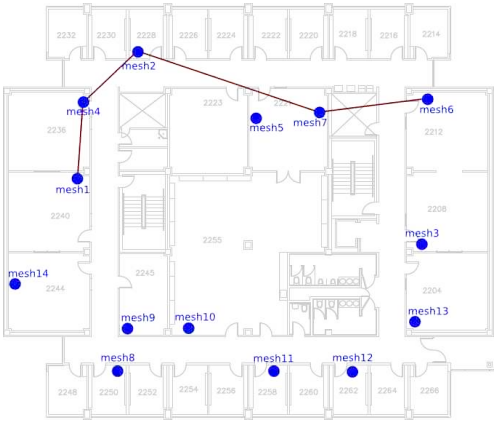


(c) 1-hop chain

Figure 4.1: Chain topologies

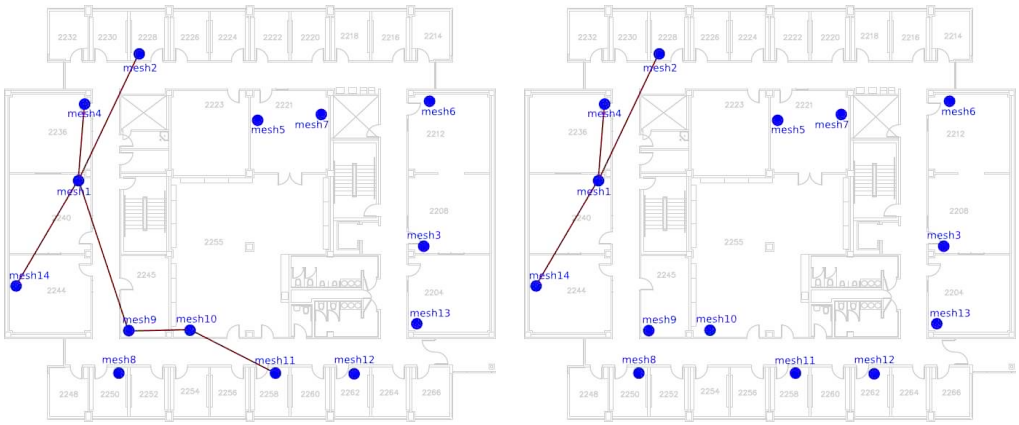


(a) 1st-floor chain



(b) 2nd-floor chain

Figure 4.2: Parallel chain topology



(a) Long asymmetric chain topology (b) Short asymmetric chain topology

Figure 4.3: Asymmetric chain topologies

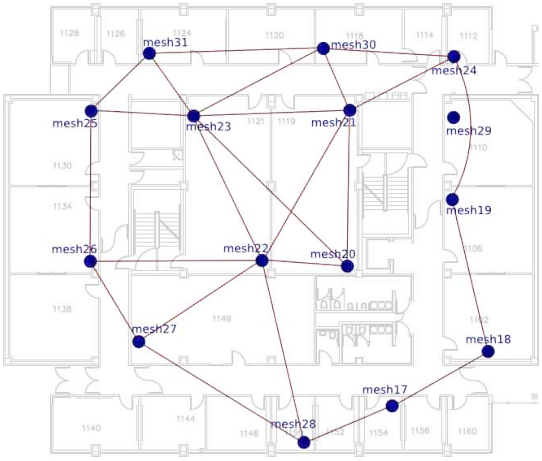


Figure 4.4: Mesh topology for random flows

4.1.3 Asymmetric Chains

The asymmetric chains are created as shown in Figs. 4.3(a) and 4.3(b). The links are set to transmission power of 8 dBm with a rate of 12 Mbps. These topologies are set up to specifically create a situation in which TCP is known to starve longer flows [18]. The long topology consists of a four-hop path from mesh4 to mesh11 and a two-hop path from mesh2 to mesh14. The short topology maintains the two-hop path but shortens the path from mesh4 to only have one hop to mesh1. In both cases, mesh1 presents the problem of a shared bottleneck between paths of different length.

4.1.4 Full Mesh

In order to emulate real traffic in a mesh network, the first floor is set up with the links shown in Fig. 4.4. Transmission power on each node is set to 10 dBm with a rate of 12 Mbps. Notice that mesh29 is not included in the mesh; this is due to technical problems with that node at experiment time.

4.2 Metrics

Transport protocol performance has typically been measured in terms of the rate of goodput and fairness. Goodput is measured by determining the amount of uncorrupted data that was transferred per unit time. Fairness is evaluated in two ways; first, with Jain's fairness metric, which is measured by inspecting the goodput of competing flows and seeing if they converge to the same rate over time when flows share a bottleneck [24]:

$$J(x_1, x_2, ..., x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} x_i^2}$$

where x_i is the goodput of flow i and n is the total number of competing flows. A Jain fairness of 1.0 indicates completely equal rates, and 0.0 would indicate complete domination of one flow over the other(s). This is most applicable when the flows share the same path, as Jain fairness optimizes for exactly equal sharing of the bandwidth.

In addition, we inspect a more general measure of log utility, defined as follows:

$$U = \begin{cases} \sum_i \log g_i & \text{if } \forall i, g_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

where g_i represents an individual flow through the topology. Instead of equal goodput rates, log utility seeks to maximize the sum of the utility functions of all flows. All utility functions are assumed to be equivalent and in a logarithmic form. In a wireless environment, this is closer to the notion of equalizing air time among the radios rather than goodput rate.

Chapter 5

Experiment Descriptions and Results

5.1 Goals

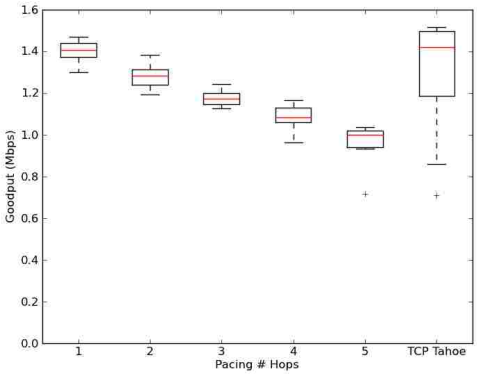
Comparisons based on goodput, Jain fairness [24], and log utility are made using the WiFu framework between TCP Tahoe, pacing, conservative windows, and delayed ACKs, as well as a combination of pacing and delayed ACKs. The parameter choices of delay heuristic for pacing, window increment for conservative windows, and number of ACKs to combine are also evaluated in order to determine the feasibility of each approach notwithstanding our specific network characteristics. These comparisons focus on the claims of TCP-AP [15], TCP-FeW [27], and TCP with Delayed ACKs [9]. We also consider the implications for current wireless network simulation tools.

5.2 General Experiment Characteristics

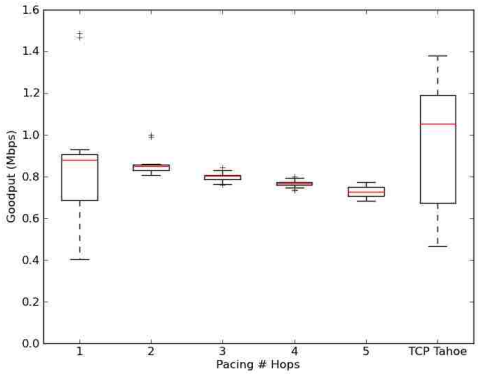
Our experiments run on the topologies defined in Section 4.1. Each experiment includes a number of experimental protocols and a TCP Tahoe control. Each of these is run for a number of iterations, with iterations of the experimental protocols interleaved with the Tahoe controls in order to minimize the effect of network conditions at different times. An FTP-like large file transfer as simulated in [15] is used to generate TCP traffic. Our transfers are one MB each. Initial evaluation of TCP-AP [15], TCP-FeW [27], and TCP with Delayed ACKs [9] is accomplished with a series of chain-topology experiments. Note that graphs include TCP Tahoe for comparison.

5.3 Pacing

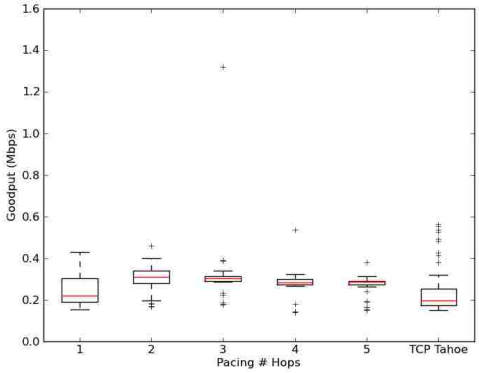
Goodput and fairness results for pacing on chain topologies are shown in Figs. 5.1 through 5.6. Goodput for pacing is generally far lower than Tahoe for all values of n, where n is the number of hops used to estimate the n-hop delay. Lower values of n tend to provide improved goodput, with



(a) 1 flow

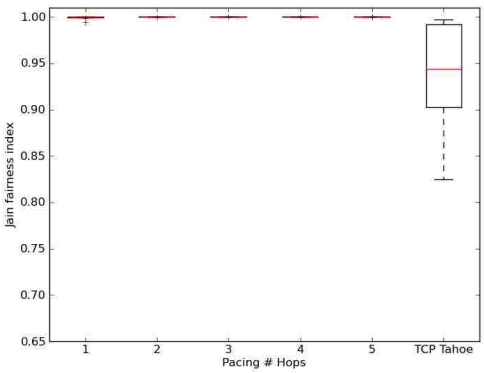


(b) 2 flows

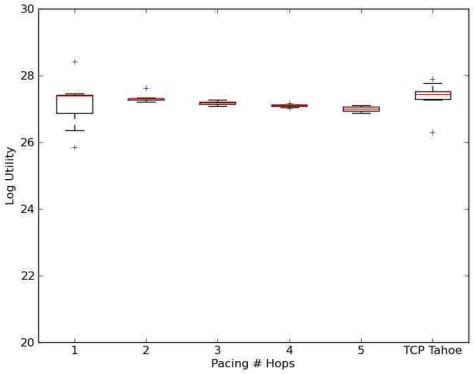


(c) 5 flows

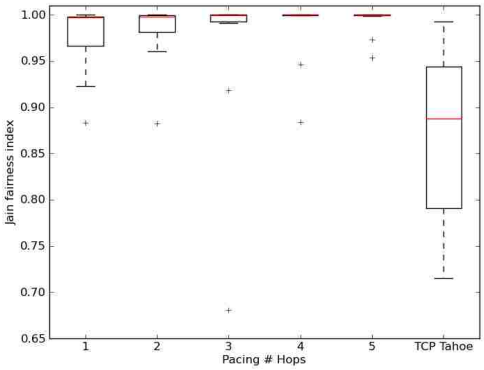
Figure 5.1: Goodput results for TCP-AP on the 9-hop chain



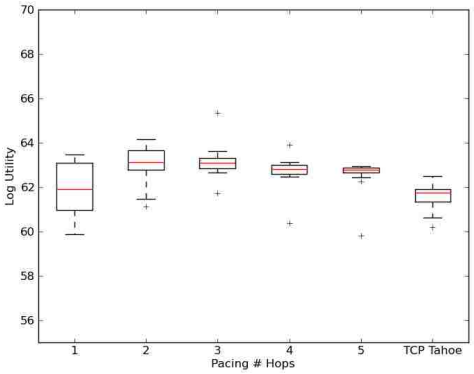
(a) Jain fairness - 2 flows



(b) Log utility - 2 flows

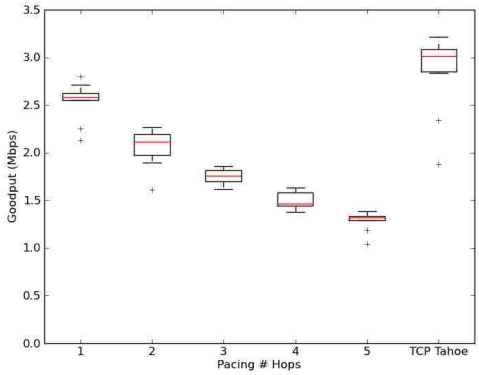


(c) Jain fairness - 5 flows

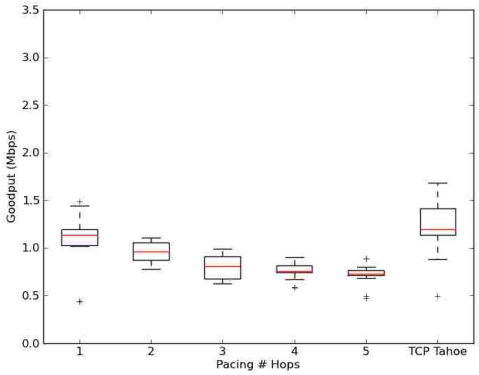


(d) Log utility - 5 flows

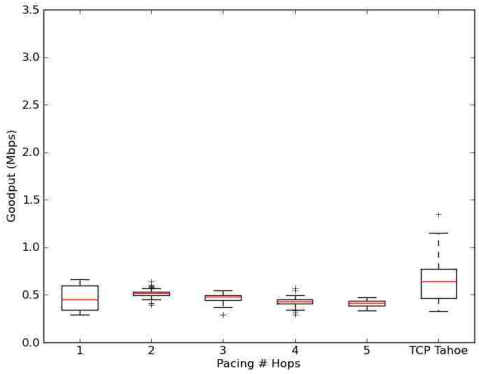
Figure 5.2: Fairness results for TCP-AP on the 9-hop chain



(a) 1 flow

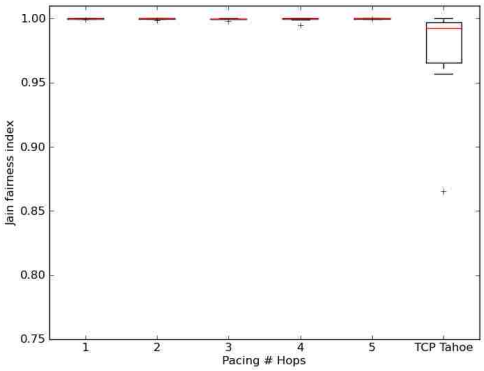


(b) 2 flows

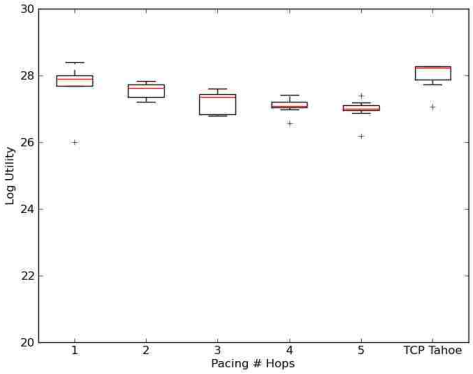


(c) 5 flows

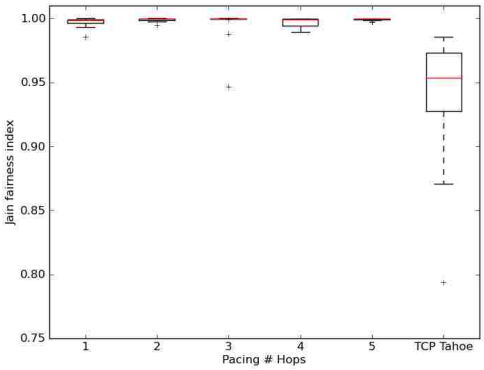
Figure 5.3: Goodput results for TCP-AP on the 4-hop chain



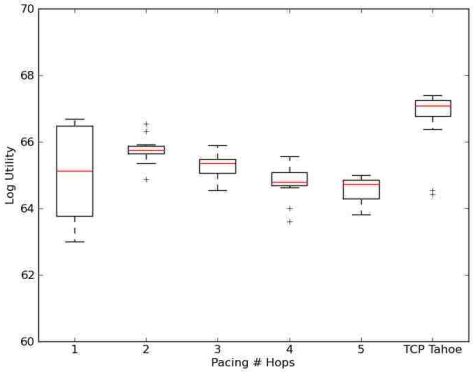
(a) Jain fairness - 2 flows



(b) Log utility - 2 flows

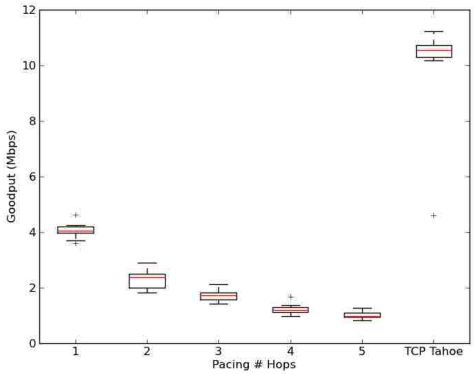


(c) Jain fairness - 5 flows

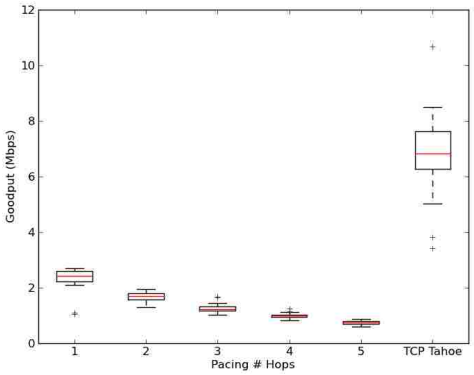


(d) Log utility - 5 flows

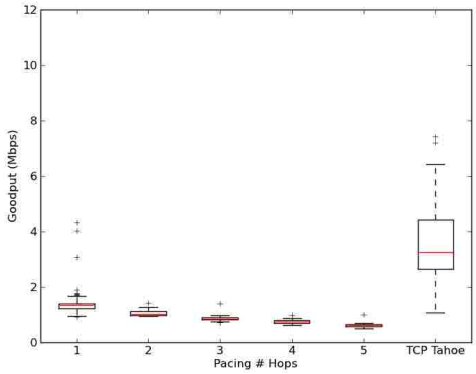
Figure 5.4: Fairness results for TCP-AP on the 4-hop chain



(a) 1 flow

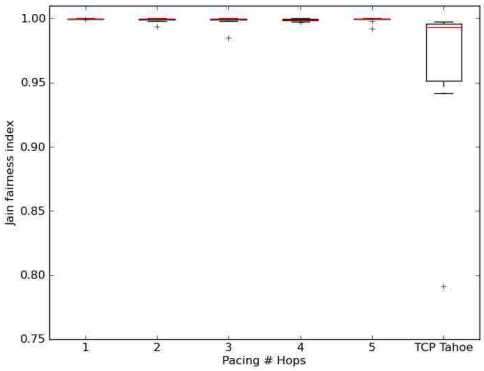


(b) 2 flows

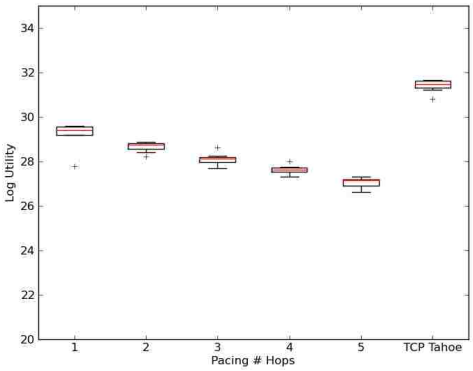


(c) 5 flows

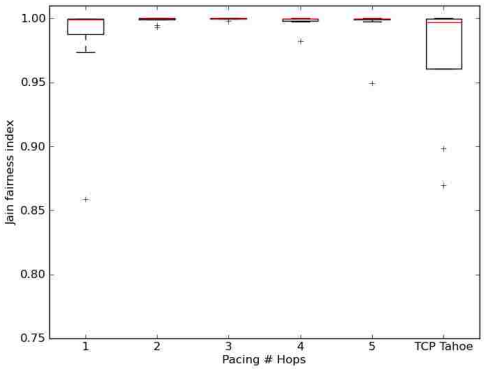
Figure 5.5: Goodput results for TCP-AP on one hop



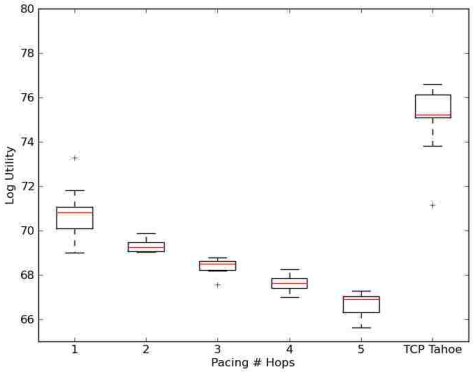
(a) Jain fairness - 2 flows



(b) Log utility - 2 flows

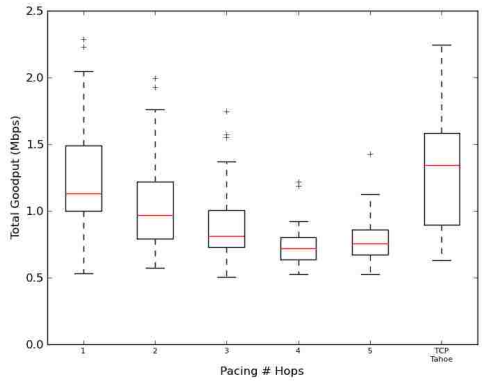


(c) Jain fairness - 5 flows

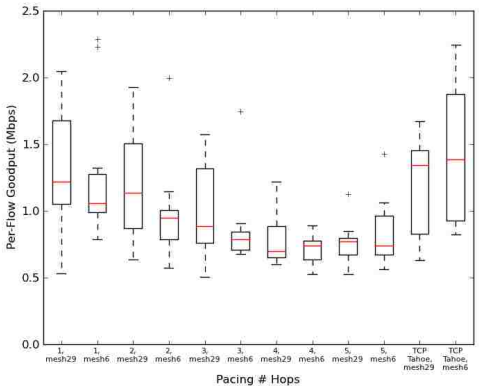


(d) Log utility - 5 flows

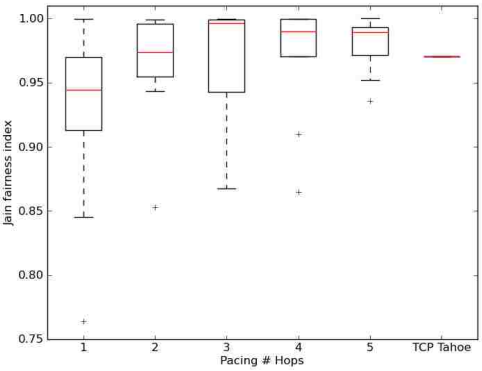
Figure 5.6: Fairness results for TCP-AP on one hop



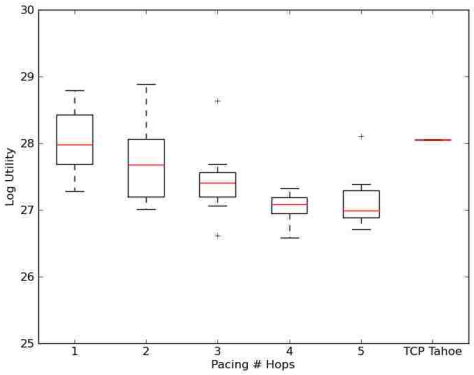
(a) Goodput



(b) Per-flow goodput



(c) Normalized Jain fairness



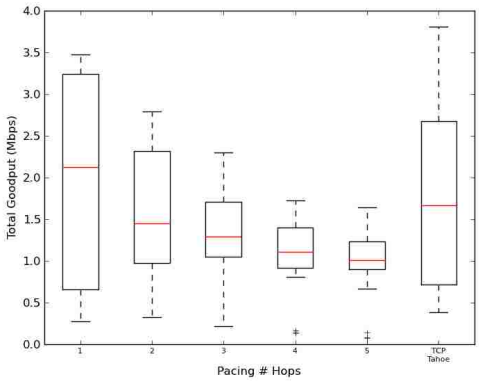
(d) Log utility fairness

Figure 5.7: Results for TCP-AP on parallel chains

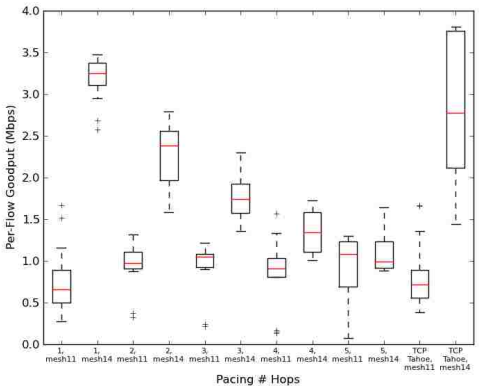
the best performance using a one-hop delay. Thus pacing provides little gain in goodput, especially over a single connection. The exception to this rule is when several flows share a long chain. Fig. 5.1(c) shows that pacing with 2- or 3-hop delay improves over both 1-hop delay and Tahoe.

Pacing shines, however, in improving Jain fairness. It performs noticeably better than Tahoe when multiple flows share a chain; see Figs. 5.2, 5.4, and 5.6. We measure this improvement using Jain's fairness index [24], which shows us that with pacing, all flows get very nearly equal rates. Pacing tends to do poorly in terms of log utility fairness because of its rather low overall goodput. Pacing thus appears to be a useful approach for improving goodput rate equality in a chain topology, but of limited use in improving overall goodput.

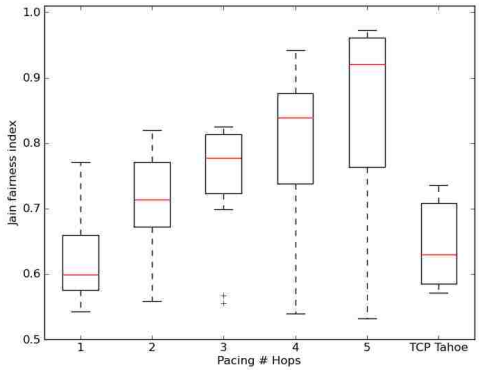
We next use a parallel chain topology to investigate the performance of pacing in the presence of interference. Goodput is reduced in this scenario; pacing achieved its best performance when n



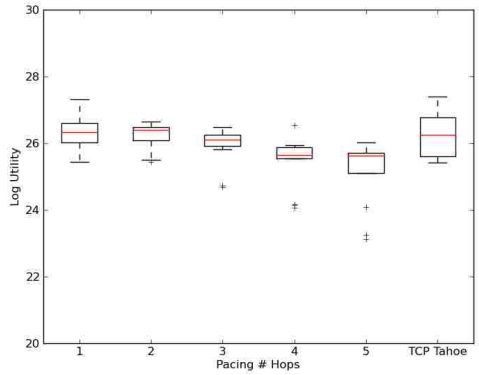
(a) Goodput



(b) Per-flow goodput



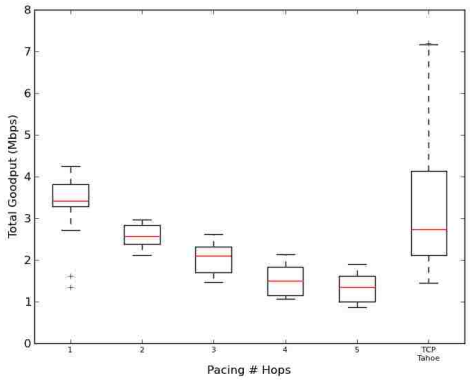
(c) Normalized Jain fairness



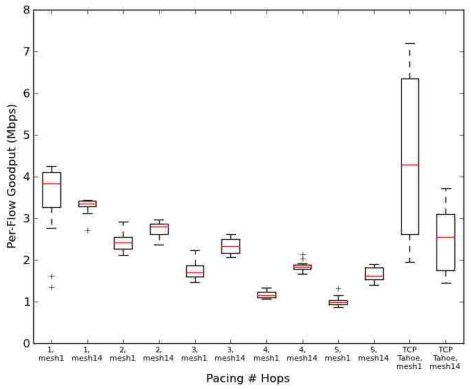
(d) Log utility fairness

Figure 5.8: Results for TCP-AP on long asymmetric chains with a bottleneck

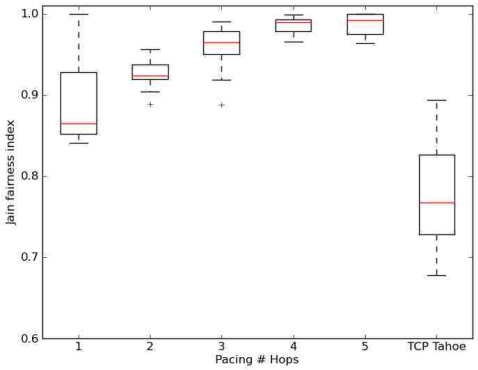
= 1, yielding the shortest inter-packet delay. Jain fairness results were not as impressive in this scenario, but still represent an improvement when $n > 1$. Per-flow goodput analysis (Fig. 5.7(b)) suggests, however, that the second-floor flow (Fig. 4.2(b)) may have been getting an increased rate at an unfair expense to the first-floor flow. Jain's index is primarily applicable when there is a shared queue in the path; the fair rate through the shared queue should be roughly equal for same-length paths. Parallel chains, however, do not share a queue; thus the rates involved could be considerably different for reasons that may or may not be measurable by Jain fairness, such as varying link quality on the completely disjoint paths. This is borne out by the log utility results, showing it is worthwhile to give the first floor path a higher goodput; both flows have higher overall goodput with Tahoe than with Pacing, even though the rate allocation is not equal.



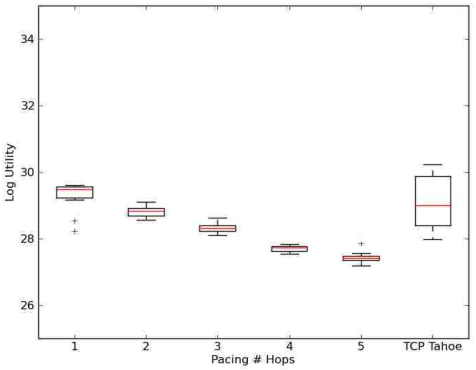
(a) Goodput



(b) Per-flow goodput



(c) Normalized Jain fairness



(d) Log utility fairness

Figure 5.9: Results for TCP-AP on short asymmetric chains

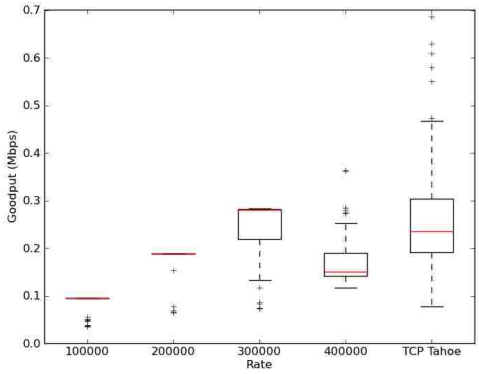
Asymmetric chains are a known weakness in TCP fairness; with a bottleneck shared between flows of uneven length, the short flow tends to starve the long flow [18]. It should be noted that we adapt Jain fairness for paths of unequal length by normalizing it according to path length; that is, we divide the average goodput of a flow by the number of hops in that flow's path. In Fig. 5.8 we see that although goodput decreases as the number of hops n increases, the reduced rate also improves normalized Jain fairness, as seen in both normalized Jain fairness and per-flow goodput; that is, the goodput rates for the competing flows are more even. Log utility is decreased because overall network utility is improved when the shorter flows get a higher rate, since they use the wireless medium more efficiently than the long flows.

As with the long asymmetric chain, the short asymmetric chain results in Fig. 5.9 show that goodput decreases with the number of hops. Per-flow goodput tends to equalize quickly on this topology because a higher rate was possible on the slower path. Normalized Jain fairness thus improves dramatically as the rate drops. As before, log utility is hampered by the falling rate of goodput.

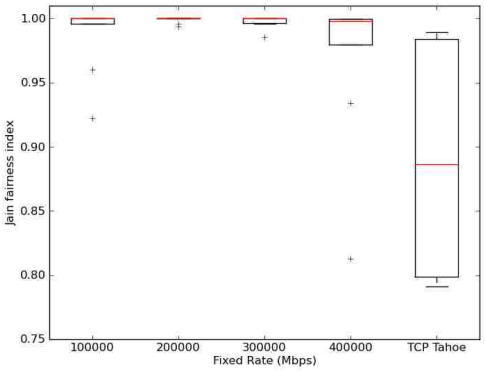
Topology	**# Flows**	**Avg. FHD (sec)**	**Rate (bps)**	**Estimate (bps)**	**% increase over AP**
9 hops	5	0.0565743	212110.33	250000	17.86
9 hops	2	0.0247930	484002.84	900000	85.95
9 hops	1	0.0140153	856206.99	1750000	104.39
4 hops	5	0.0430432	278789.60	400000	43.48
4 hops	2	0.0199717	600849.84	1400000	133.00
4 hops	1	0.0117432	1021863.19	2750000	169.12
One hop	5	0.0246570	486676.07	2750000	465.06
One hop	2	0.0157963	759670.99	7000000	821.45
One hop	1	0.0116968	1025918.37	11000000	972.21

Table 5.1: Rate experiment results

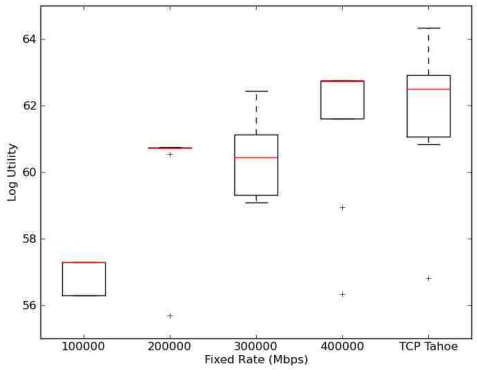
If we want to maximize goodput or log utility, the n-hop delay metric is clearly too conservative, but the benefit that pacing achieves with respect to Jain's fairness metric merits further investigation. Accordingly, we run some experiments on our 9-hop, 4-hop, and one-hop chain topologies with various fixed rates in an effort to determine if some more correct rate would allow us to have goodput as good or better than Tahoe while still maintaining improved Jain fairness properties. Our experiments use rates ranging from .1 to .4 Mbps and from 1 to 5 Mbps, but not



(a) Goodput



(b) Jain fairness



(c) Log utility fairness

Figure 5.10: Goodput results for fixed-rate pacing with five flows on the 9-hop chain

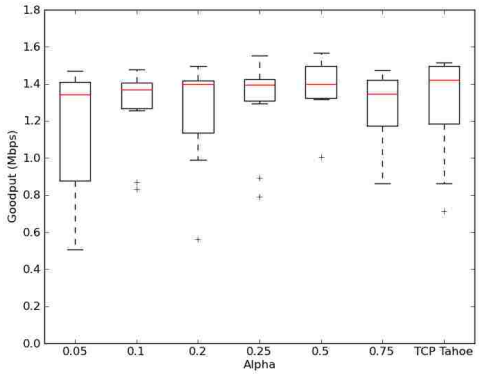
all results are shown. Fig. 5.10 is an example of the results we observe. Note that as the rate we set approaches or goes above the mean rate achieved by Tahoe, the variance among the rate-limited flows begins to grow; however, near and even slightly above the mean rate that Tahoe achieves, we still have an apparent Jain fairness benefit indicated by lower variance. In our Jain fairness analysis in Fig. 5.10(b), we see that Jain fairness is maximized somewhere between 200 kbps and 300 kbps. The mean goodput for TCP Tahoe on this topology was about 250 kbps, and so it appears that a metric which accurately estimates the mean TCP goodput could be used with pacing to improve Jain fairness in a chain topology while maintaining higher goodput.

The full goodput results are seen in Table 5.1. We show our rate comparison of TCP-AP and an estimate based on the mean rate achieved by TCP Tahoe. In the first two columns, we define the topology and number of flows involved in one particular experiment. The average actual four-hop delay calculated by TCP-AP and its corresponding rate in bits per second are shown in the third and fourth columns. The penultimate column presents our estimates of the optimal rate based on our Jain fairness observation and on our fixed-rate experiments. The difference between the FHD estimate and our estimate and the percentage by which the average Tahoe rate exceeded the FHD-based rate are shown in the final two columns.

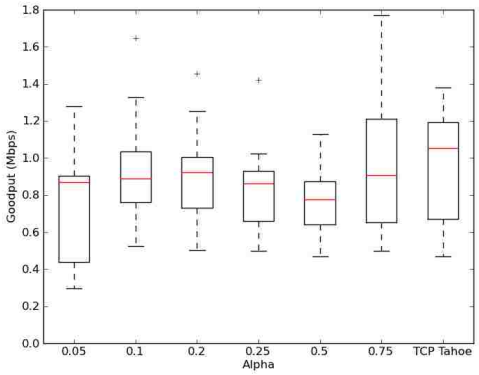
Notice here that the 4-hop delay metric consistently underestimated the optimal rate by a large factor. This effect is most extreme in low-loss chain scenarios; with a single flow over a single-hop path, Tahoe converged to an average rate nearly 1000% higher than the FHD-based rate. For low-loss chains, TCP Tahoe appears to find the optimal rate overall; however, the bursty traffic causes unfair sharing of that average rate. We find that n-hop delay is an incorrect metric to rely on for pacing flows, but future work on an improved metric to estimate mean TCP goodput could result in a very fair rate-limited version of TCP.

5.4 Conservative Windows

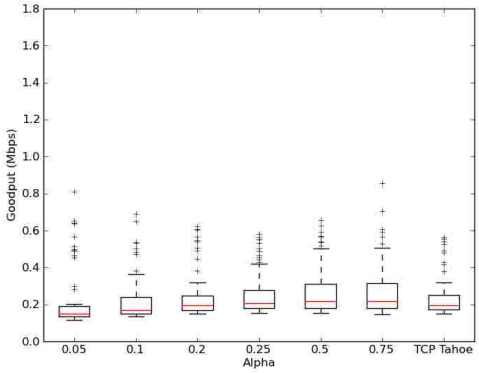
Results for conservative windows (in TCP-FeW [27]) are uneven. On chain topologies, conservative windows of any size tend to have mean goodputs close to what Tahoe achieves, with occasional improvements in Jain fairness. We do not observe any indication that any value for the scaling



(a) 1 flow

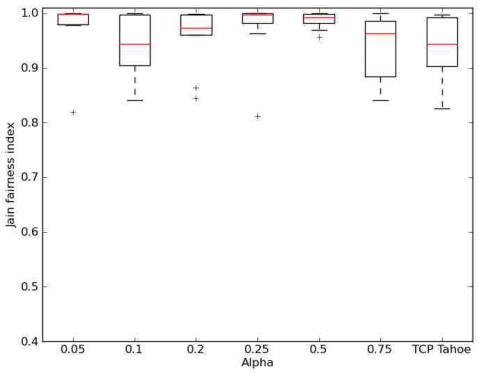


(b) 2 flows

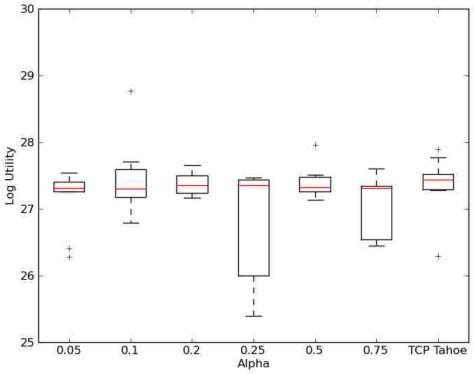


(c) 5 flows

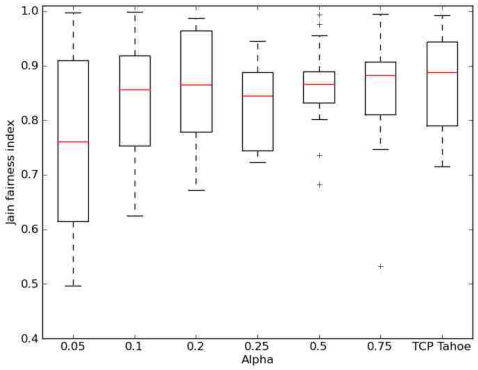
Figure 5.11: Goodput results for TCP-FeW on the 9-hop chain



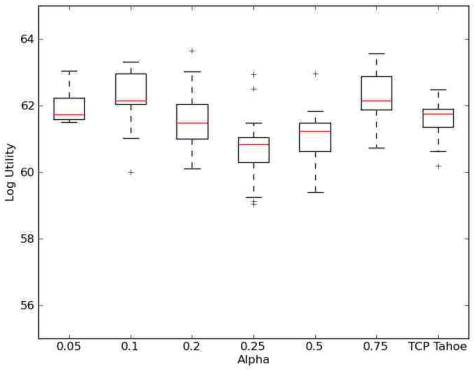
(a) Jain fairness - 2 flows



(b) Log utility - 2 flows

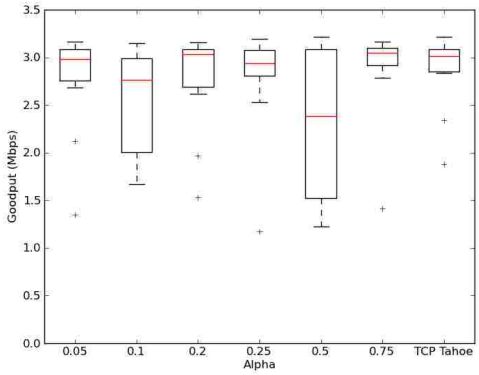


(c) Jain fairness - 5 flows

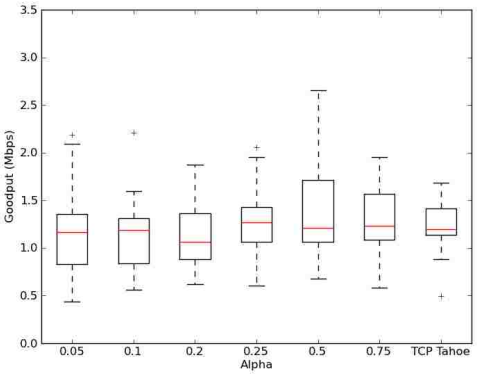


(d) Log utility - 5 flows

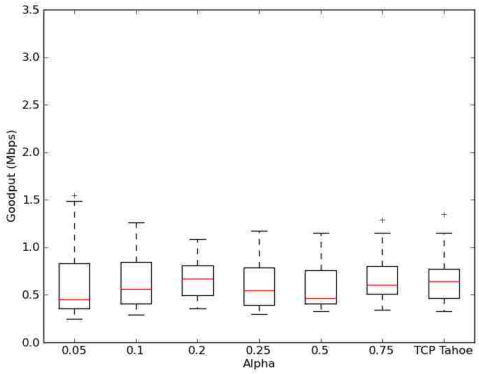
Figure 5.12: Fairness results for TCP-FeW on the 9-hop chain



(a) 1 flow

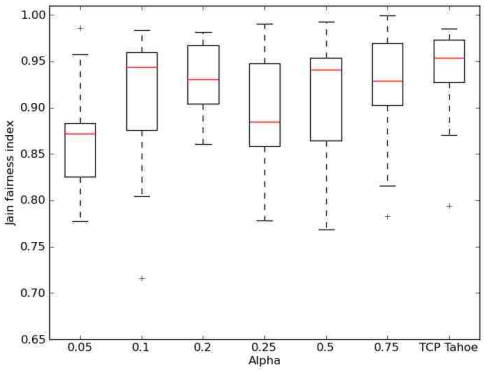


(b) 2 flows

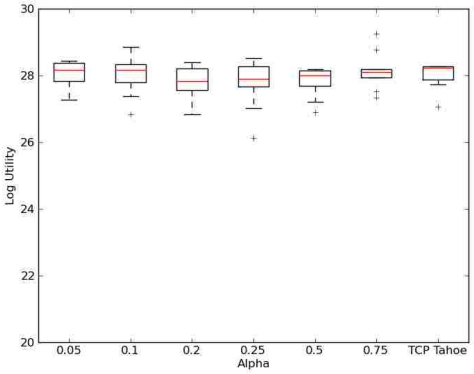


(c) 5 flows

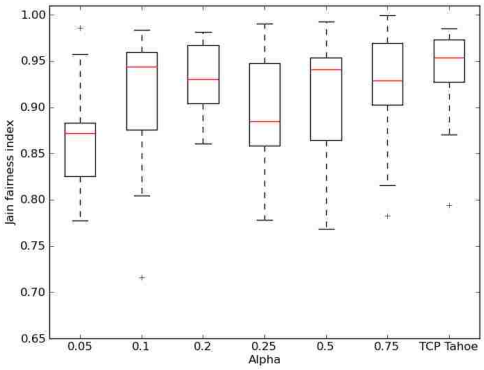
Figure 5.13: Goodput results for TCP-FeW on the 4-hop chain



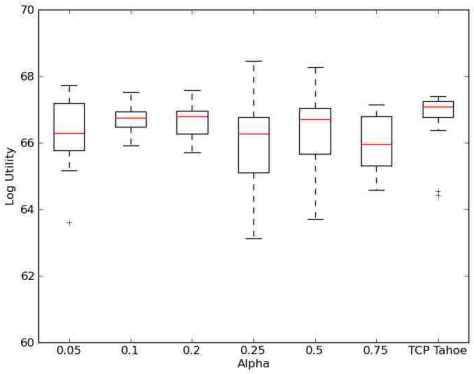
(a) Jain fairness - 2 flows



(b) Log utility - 2 flows

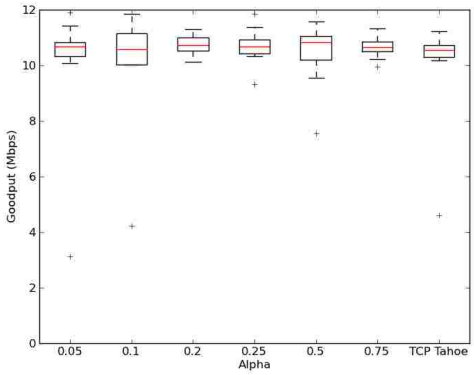


(c) Jain fairness - 5 flows

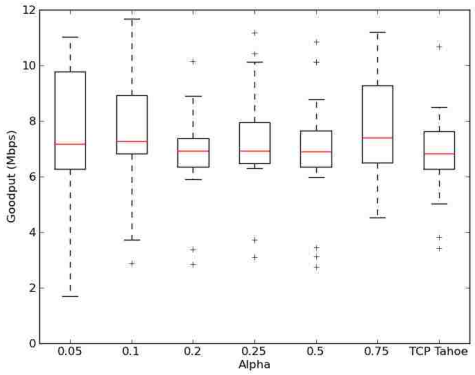


(d) Log utility - 5 flows

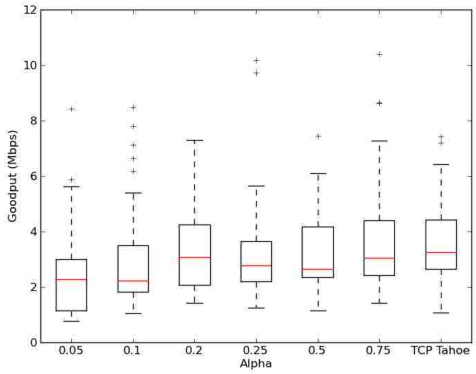
Figure 5.14: Fairness results for TCP-FeW on the 4-hop chain



(a) 1 flow

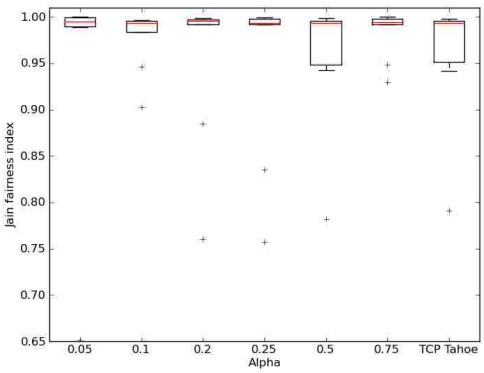


(b) 2 flows

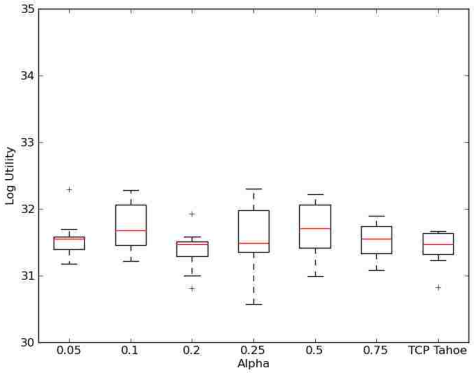


(c) 5 flows

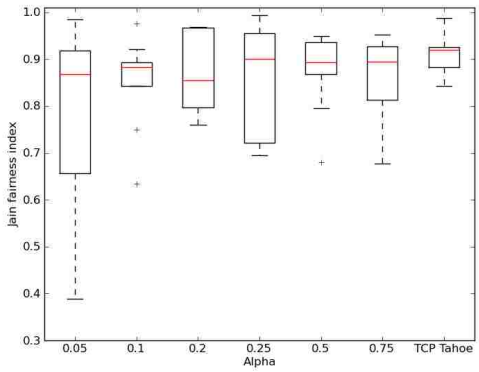
Figure 5.15: Goodput results for TCP-FeW on one hop



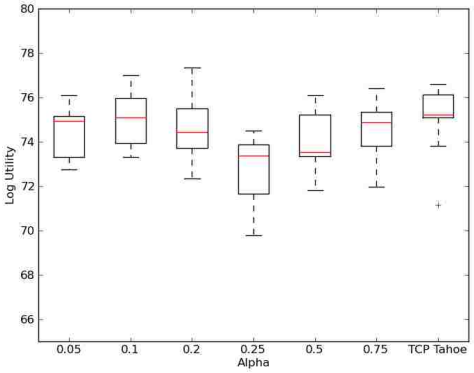
(a) Jain fairness - 2 flows



(b) Log utility - 2 flows



(c) Jain fairness - 5 flows



(d) Log utility - 5 flows

Figure 5.16: Fairness results for TCP-FeW on one hop

factor α is best, nor any conclusive evidence that TCP-FeW provides consistent improvement to Tahoe's performance.

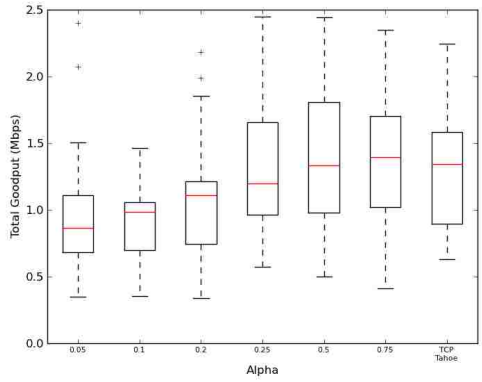
The 9-hop chain topology results in Fig. 5.11 show that Tahoe's mean goodput is higher than that of any value of α for one or two simultaneous flows. For five simultaneous flows, mean goodput was higher for $\alpha = 0.5$ or $\alpha = 0.75$, but only by a very small amount. Fairness is extremely inconsistent; Fig. 5.12 shows every value of α improving on TCP Tahoe's Jain fairness when two flows compete, but no α value improves Jain fairness when five flows compete. Log utilities are similarly inconclusive, staying largely below Tahoe's performance for two flows and improving slightly for five flows when $\alpha > 0.2$.

Mean goodput for TCP-FeW on the 4-hop topology tends to be under that of TCP Tahoe in Fig. 5.13. The only exceptions are slight improvements for two flows, $\alpha = 0.25$ and five flows, $\alpha = 0.2$. Jain fairness (Fig. 5.14) is consistently below Tahoe's performance, and log utility only is improved for two flows with $\alpha > 0.2$.

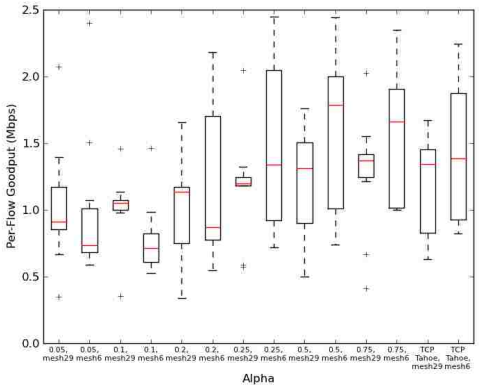
TCP-FeW on the one-hop topology does show slightly improved goodput for one and two flows in Fig. 5.15. This holds for all values of α. For five flows, however, all values of α underperform Tahoe in mean goodput. Fairness results on one hop are seen in 5.16. For two flows, Jain fairness hovers around the value for Tahoe, staying within ± 0.02. Five-flow Jain fairness, however, stays well below Tahoe's performance. Log utility rises and falls with the corresponding values of mean goodput seen in Fig. 5.15.

On parallel chains (Fig. 5.17) conservative windows provide no significant improvement to either goodput or Jain fairness over standard TCP Tahoe. Log utility fairness is slightly improved for high values of α, but the improvement is too small to be of great concern, especially since it occurs as α increases and makes TCP-FeW more closely approximate TCP Tahoe behavior.

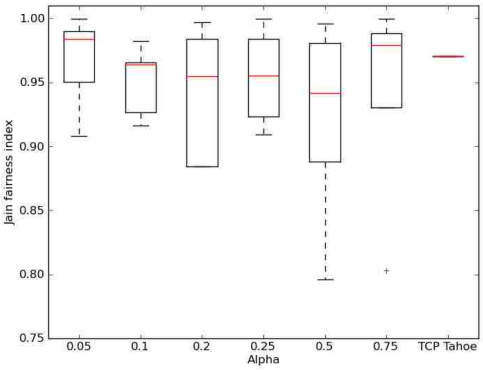
The final experiments we run with conservative windows use the asymmetric chain topologies (Figs. 5.18 and 5.19). As in previous experiments, the results do not show any significant improvement over TCP Tahoe with the exception of an improvement in normalized Jain fairness on the shorter chain in Fig. 5.19(c). Goodput does appear to be improved in some cases, but the improvement is too slight to be statistically significant. The highest achieved Jain fairness of 0.83 is not very impressive, however, and it appears that pacing is a much better approach to ensuring equal goodput rates (see Section 5.3).



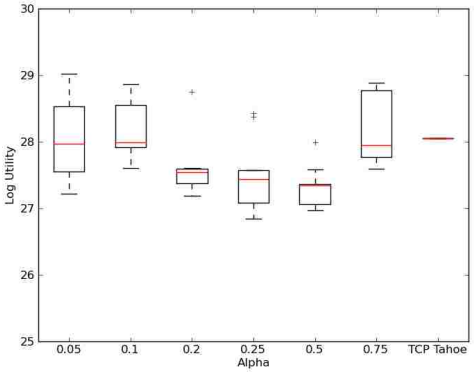
(a) Goodput



(b) Per-flow goodput

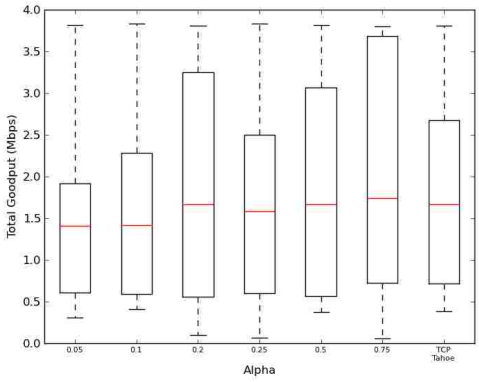


(c) Normalized Jain fairness

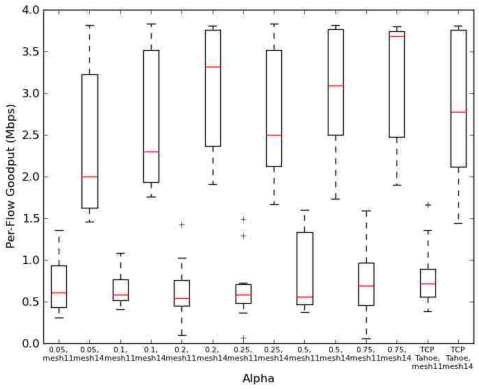


(d) Log utility fairness

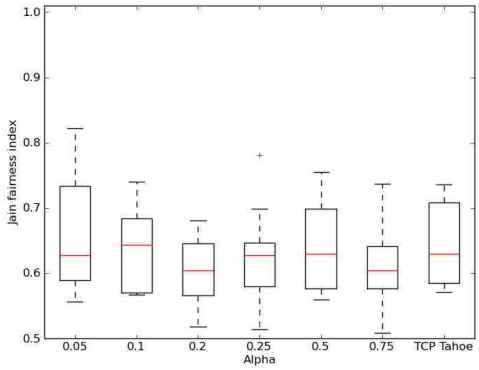
Figure 5.17: Results for TCP-FeW on parallel chains



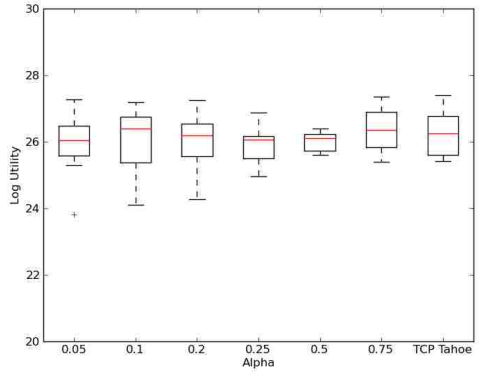
(a) Goodput



(b) Per-flow goodput

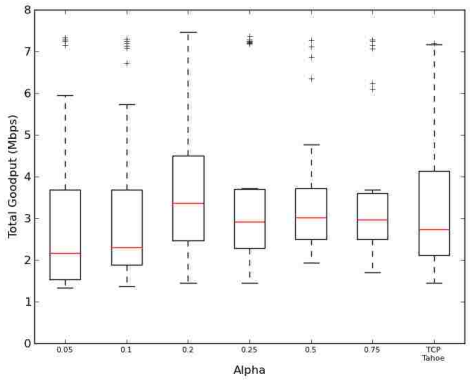


(c) Normalized Jain fairness

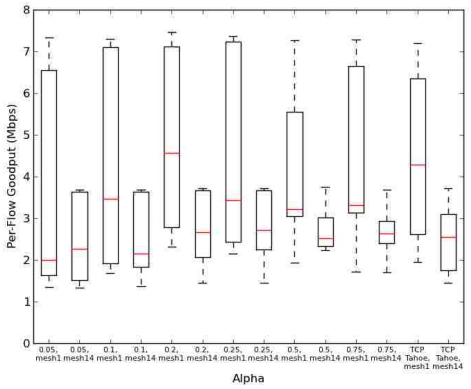


(d) Log utility fairness

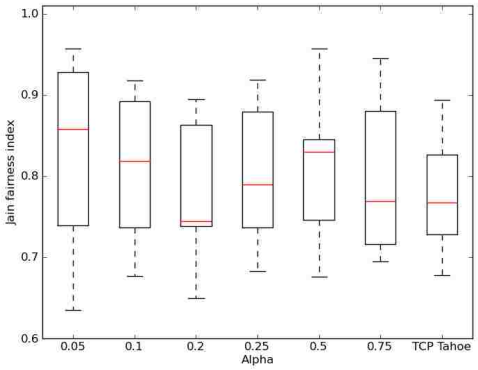
Figure 5.18: Results for TCP-FeW on long asymmetric chains



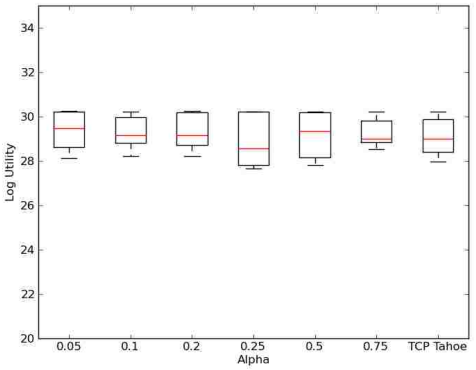
(a) Goodput



(b) Per-flow goodput

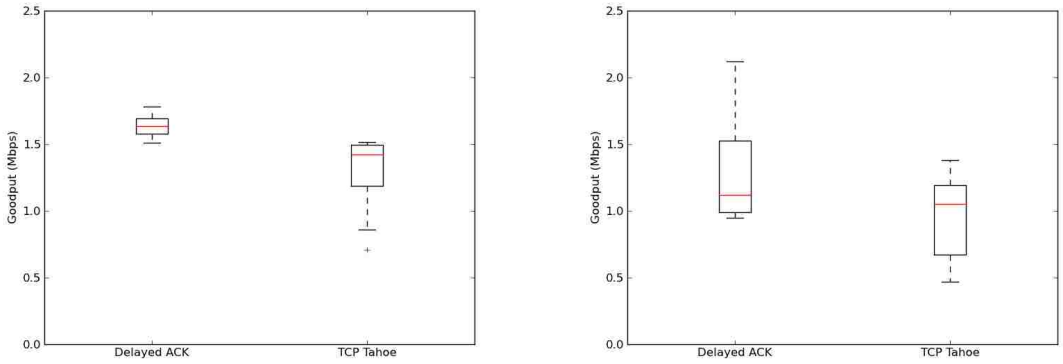


(c) Normalized Jain fairness



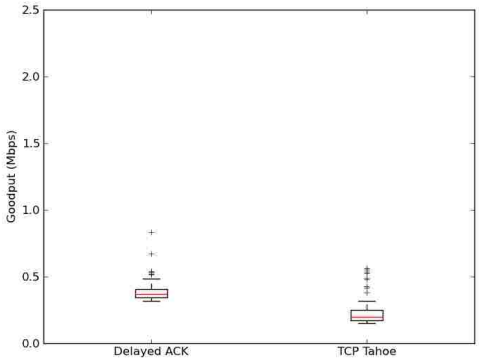
(d) Log utility fairness

Figure 5.19: Results for TCP-FeW on short asymmetric chains



(a) 1 flow

(b) 2 flows



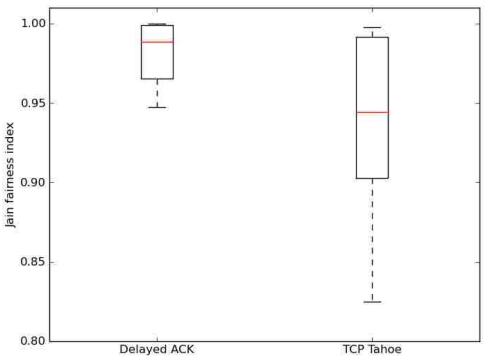
(c) 5 flows

Figure 5.20: Goodput results for delayed ACKs on the 9-hop topology

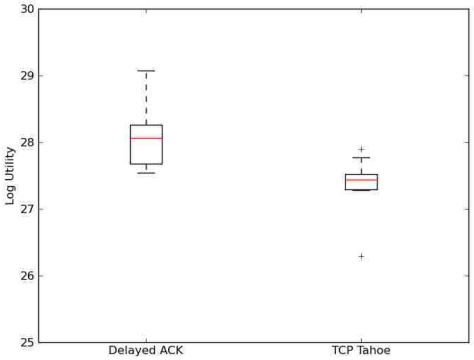
It should be noted that TCP-FeW was developed for *mobile* wireless mesh networks with dynamic routing. It is possible that these two additional factors create an environment where conservative windows are helpful, and this possibility should be investigated in future work. For the purposes of non-mobile, statically-routed wireless mesh networks, however, conservative windows do not present any notable improvement.

5.5 Delayed ACKs

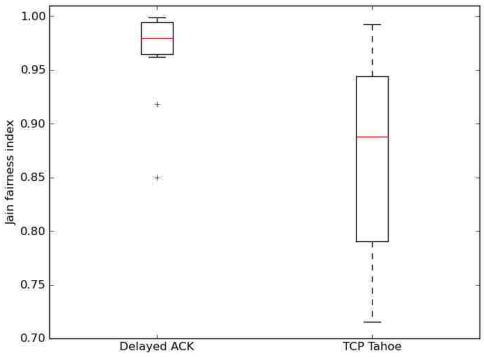
Delayed ACKs have a remarkable goodput increase on each of the chain topologies, and the log utilities are therefore also significantly higher than TCP Tahoe; however, the advantage decreases as more flows compete. The two-flow case in Fig. 5.20(b) and the five-flow case in Fig. 5.24(c) have the lowest mean goodput increase over Tahoe, but they do shift the upper and lower bounds of the



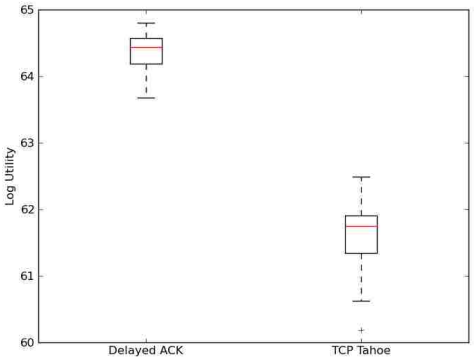
(a) Jain fairness - 2 flows



(b) Log utility - 2 flows

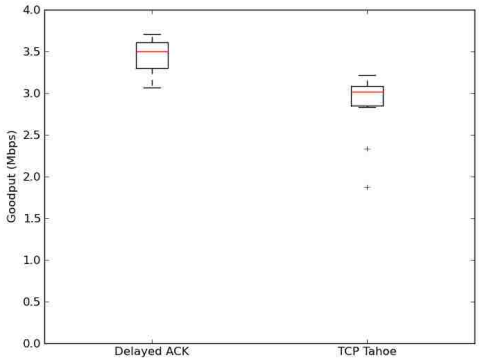


(c) Jain fairness - 5 flows

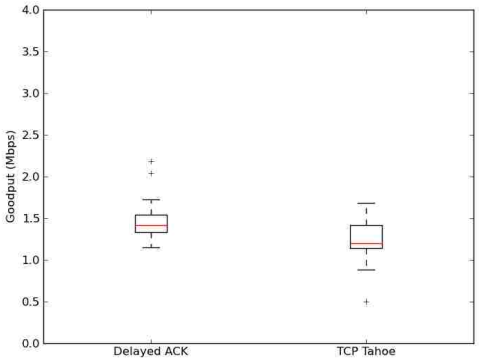


(d) Log utility - 5 flows

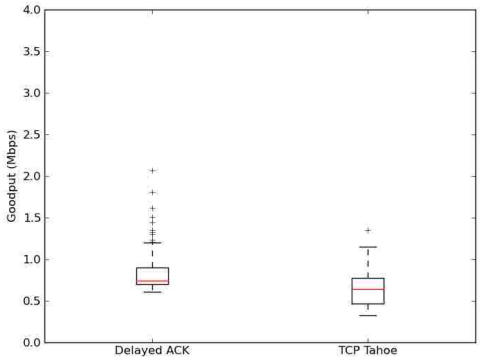
Figure 5.21: Fairness results for Delayed ACKs on the 9-hop chain



(a) 1 flow

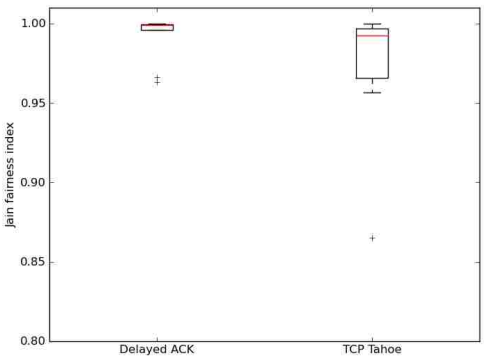


(b) 2 flows

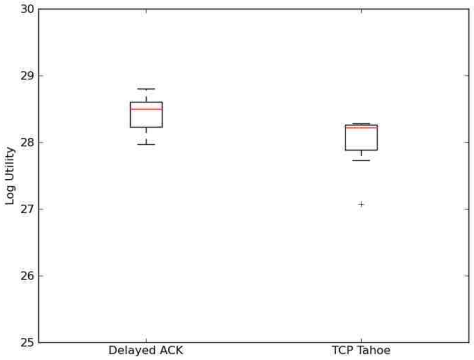


(c) 5 flows

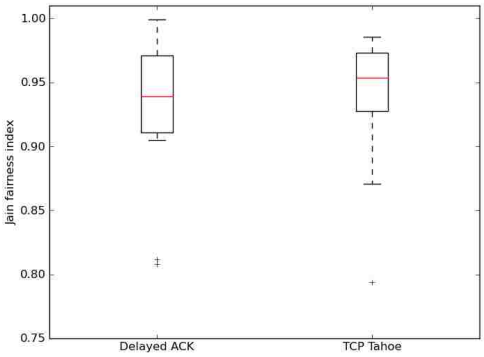
Figure 5.22: Goodput results for delayed ACKs on the 4-hop topology



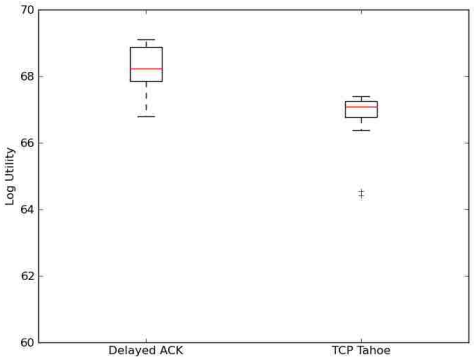
(a) Jain fairness - 2 flows



(b) Log utility - 2 flows

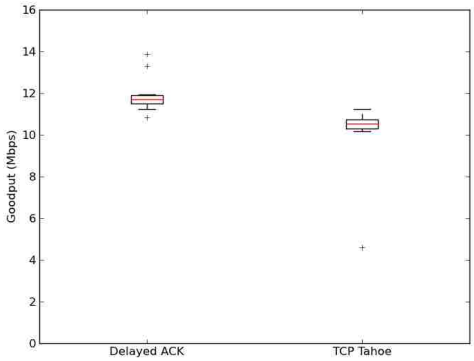


(c) Jain fairness - 5 flows

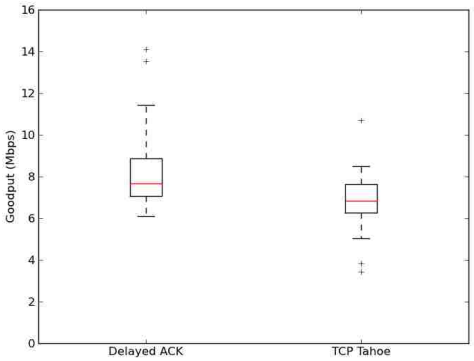


(d) Log utility - 5 flows

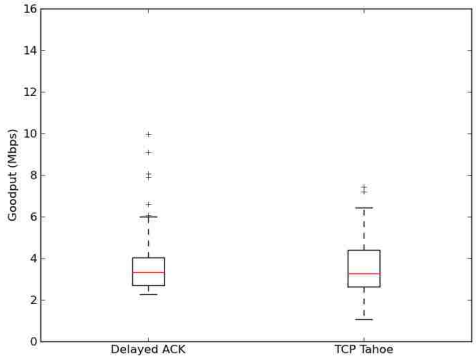
Figure 5.23: Fairness results for Delayed ACKs on the 4-hop chain



(a) 1 flow

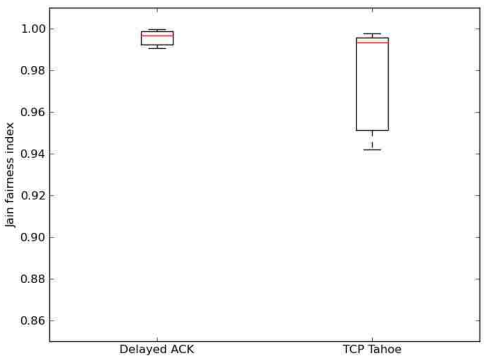


(b) 2 flows

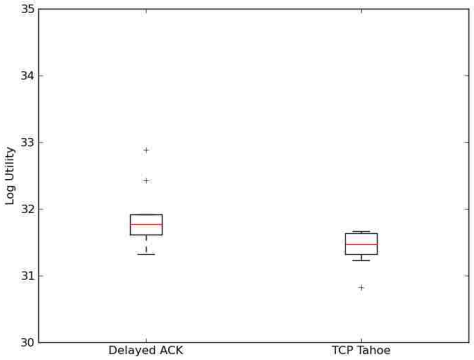


(c) 5 flows

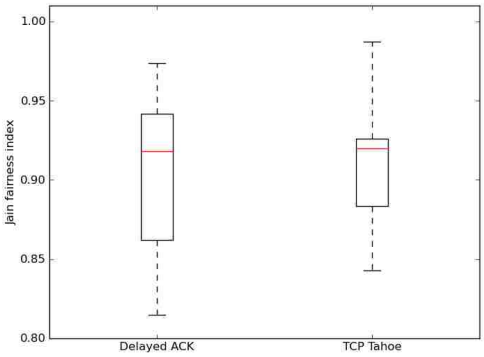
Figure 5.24: Goodput results for delayed ACKs on the one-hop topology



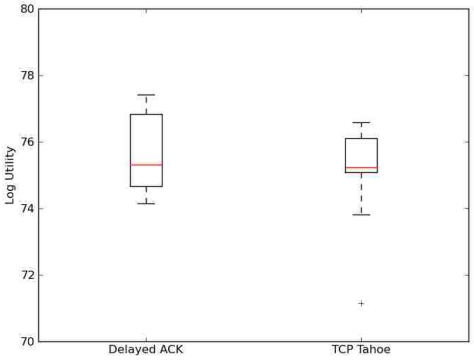
(a) Jain fairness - 2 flows



(b) Log utility - 2 flows

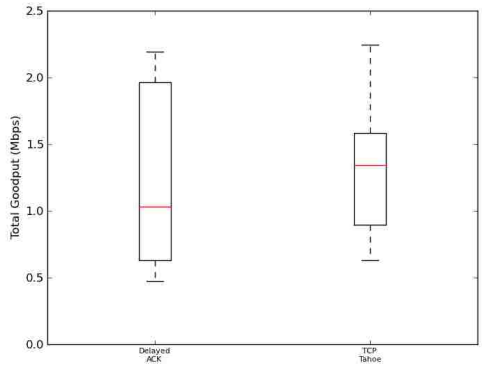


(c) Jain fairness - 5 flows

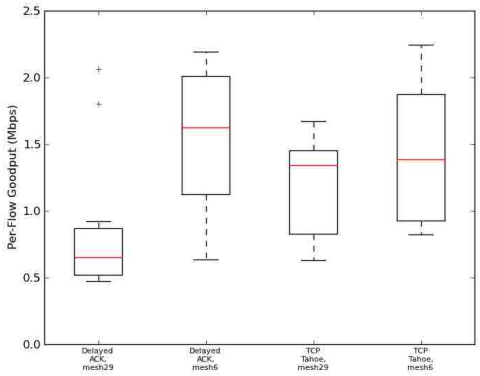


(d) Log utility - 5 flows

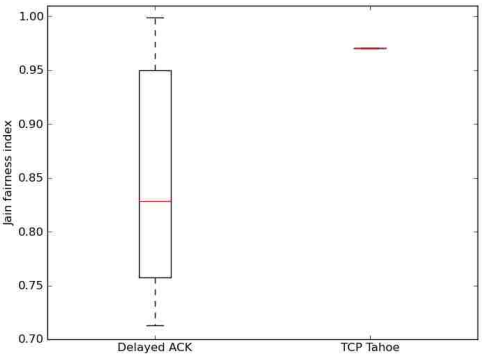
Figure 5.25: Fairness results for Delayed ACKs on the one-hop chain



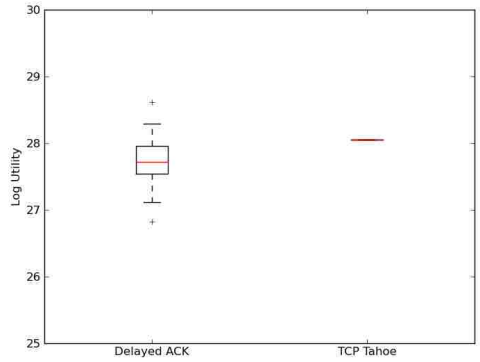
(a) Goodput



(b) Per-flow goodput



(c) Normalized Jain fairness

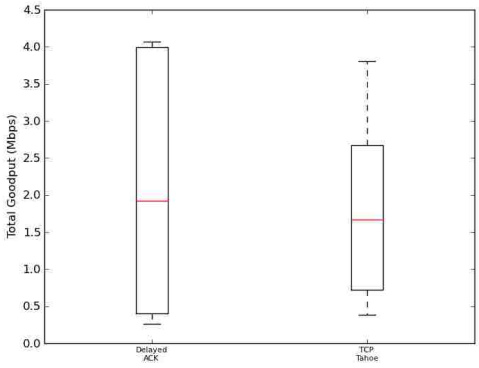


(d) Log utility fairness

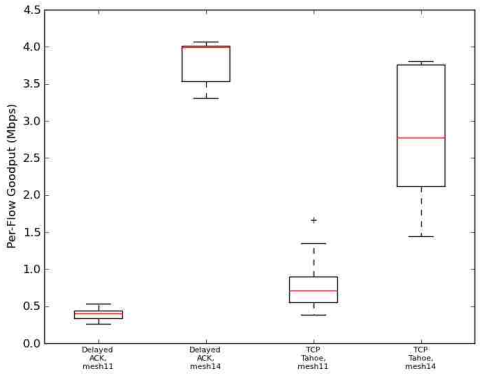
Figure 5.26: Results for delayed ACKs on parallel chains

distribution of flows higher. Jain fairness in Figures 5.23 and 5.25 are comparable to Jain fairness for TCP Tahoe, and Jain fairness on the long chain even improves on Tahoe's performance (Fig. 5.21).

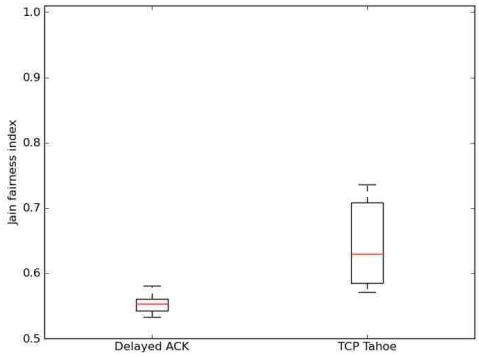
Our experiments on parallel chains show a weakness in Delayed ACKs; the interference caused by the parallel chains causes increased packet loss rates for each flow as seen in Fig. 5.26(a). The loss of one delayed ACK has a much greater effect on TCP congestion control than the loss of normal ACKs. Consequently, the average goodput in a high-interference situation suffers more for TCP with delayed ACKs than for normal TCP. The per-flow goodput results in Figure 5.26(b) confirm that the effects of this loss are unfairly distributed between flows, which also causes the Jain fairness to drop below that of Tahoe in Figure 5.26(c). Log utility fairness also drops because of both the decrease in fairness and the general drop in goodput.



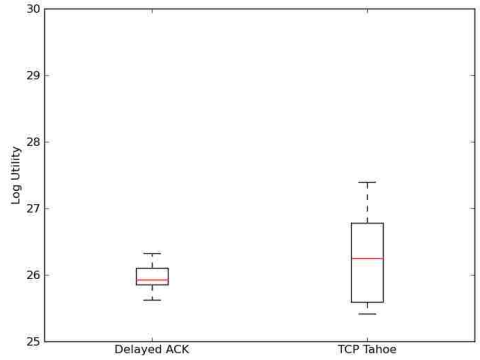
(a) Goodput



(b) Per-flow goodput

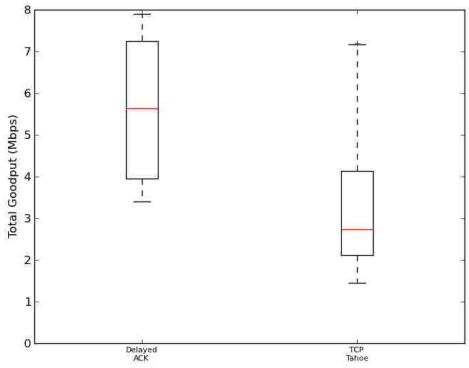


(c) Normalized Jain fairness

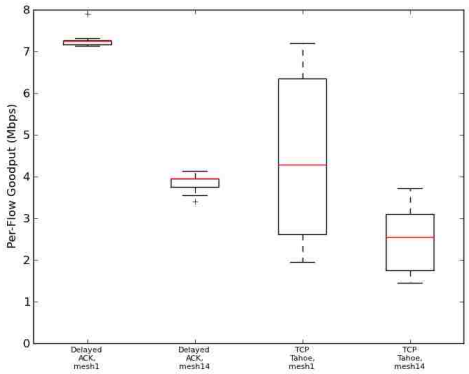


(d) Log utility fairness

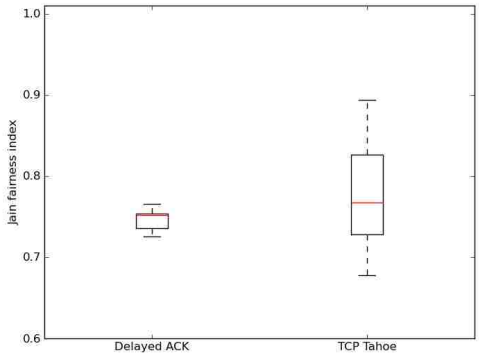
Figure 5.27: Results for delayed ACKs on the long asymmetric topology



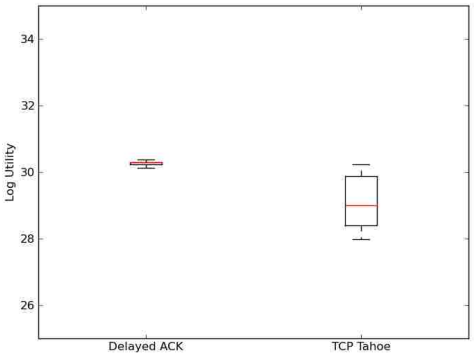
(a) Goodput



(b) Per-flow goodput



(c) Normalized Jain fairness



(d) Log utility fairness

Figure 5.28: Results for delayed ACKs on the short asymmetric topology

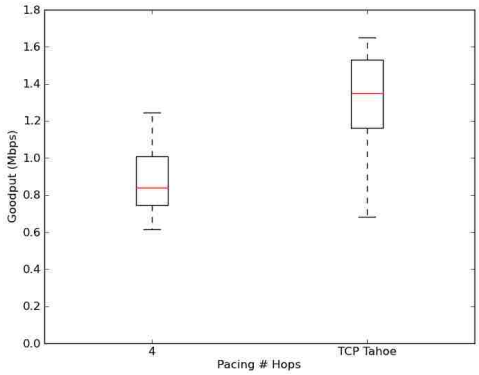
The experiments on asymmetric chains again show the goodput benefit of delayed ACKs. The goodput results in Fig. 5.27 show that average goodput is dramatically increased over TCP Tahoe. Fairness, however, is extremely poor; the per-flow analysis of long asymetric chains in Fig. 5.27(b) shows that one flow had 4 Mbps while the other was stuck at 0.5 Mbps, a difference of a factor of eight. TCP Tahoe had a disparity between the flows of a factor of four, from 3 Mbps to 0.75 Mbps; we see that delayed ACKs can exacerbate existing fairness problems in TCP. The short chains demonstrate an interesting phenomenon, however; in Fig. 5.28(b) we notice that although the improved goodput was not fairly doled out, the flow with lower goodput was still performing above the mean goodput of TCP Tahoe in Fig. 5.28(a).

We conclude that the reduction in ACK traffic is generally effective in improving TCP goodput on wireless channels. A particularly lossy environment such as with parallel chains competing for the wireless medium can cause drops in goodput due to the significant increase in effect of a dropped ACK, but we find that taking this risk may be an overall improvement in a wireless mesh.

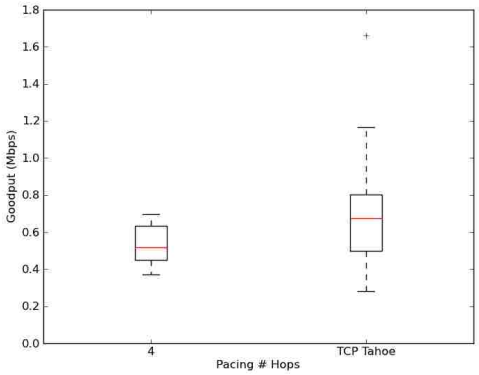
5.6 Pacing with Delayed ACKs

Because paced flows are shown to improve fairness and delayed ACKs are shown to improve goodput, we perform several experiments to determine if the combination of those two approaches gives us a more significant benefit. The authors of TCP-AP [15] implemented their simulation incorporating both their four-hop delay metric and dynamically delayed ACKs as described in Section 3.6. We opt not to test other values of n for the n-hop delay except for $n = 1$ on the one-hop chain topology because higher values of n result in similar Jain fairnesses in Section 5.3. We choose $n = 4$ in particular because it also replicates the same parameters used by the authors of TCP-AP. We do include $n = 1$ on the one-hop chain topology because there is only one hop on the path, so a four-hop delay estimate may be inappropriate. We use WiFu-End to create a combination protocol using the reliability module of dynamic delayed ACKs and the rate controller of our TCP-AP implementation.

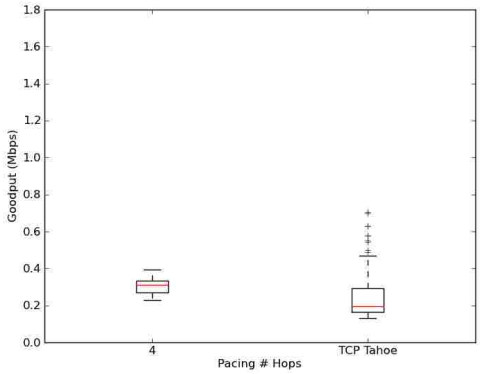
We begin with chain experiments to test goodput improvement. The 9-hop chain results in Fig. 5.29 are mixed; the goodput-limiting effect of pacing often overcomes the goodput-improving effect of delayed ACKs. This means that no matter what benefit dynamically delayed ACKs may



(a) 1 flow

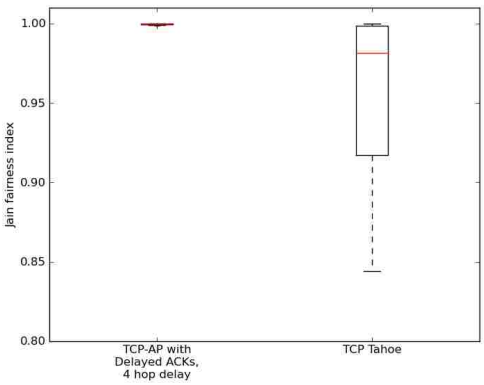


(b) 2 flows

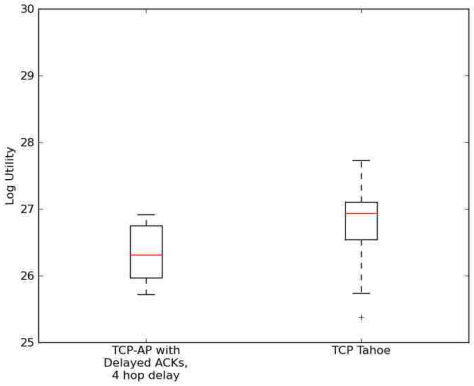


(c) 5 flows

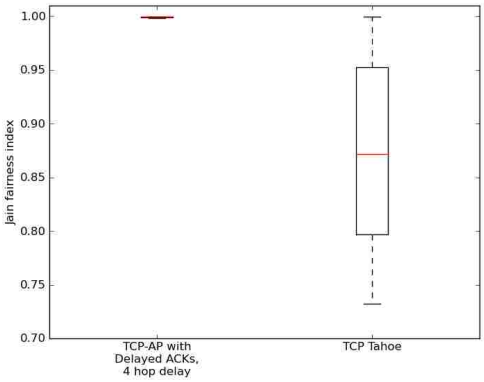
Figure 5.29: Goodput results for combined TCP-AP and Delayed ACKs on the 9-hop chain



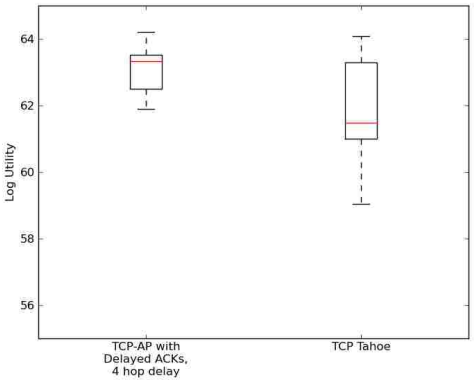
(a) Jain fairness - 2 flows



(b) Log utility - 2 flows



(c) Jain fairness - 5 flows



(d) Log utility - 5 flows

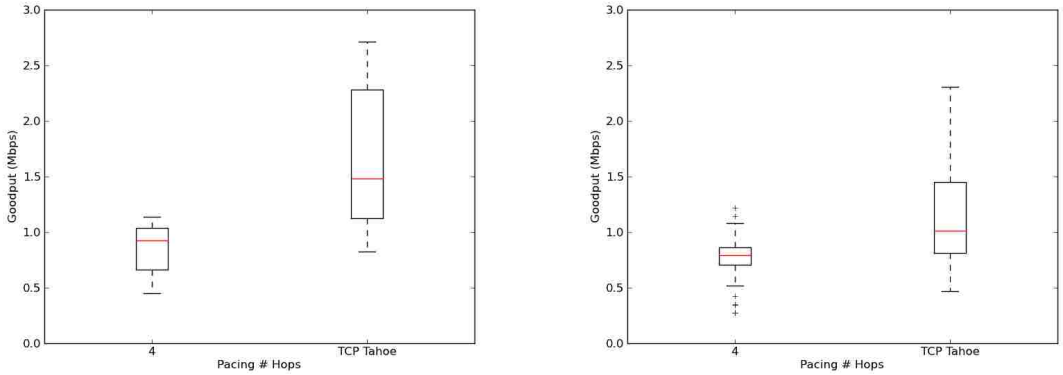
Figure 5.30: Fairness results for combined TCP-AP and Delayed ACKs on the 9-hop chain

provide, it cannot be seen due to the low rate-limit. Additionally, there may be little reduction in ACK traffic because the delayed ACK timer fires before a full complement of data packets to ACK has arrived. This is most likely to occur under heavy traffic conditions; the rates estimated in Table 5.1 show that the average calculated delay between packets was over .04 seconds for five flows on both 9- and 4-hop chains. Such a rate would only permit at most two data packets to arrive within the .1 second time limit imposed by delayed ACKs. Our experiment on this chain with five simultaneous flows, as seen in Fig. 5.29(c), shows an increase in goodput over Tahoe because the optimal shared fair rate is reduced to a rate closer to the four-hop-delay estimate. Fairness is dramatically improved, with the Jain fairness results in Fig. 5.30 significantly higher than Tahoe, which is consistent with our findings on pacing in Section 5.3. Log utility is improved for the five-flow case because both goodput and Jain fairness showed improvement, but the two-flow case in Fig. 5.30(b) shows that the decrease in goodput has a large effect on log utility.

Our experiments on the 4-hop chain (Fig. 5.31) also confirm prior experiments; the average goodput for one and two flows is decreased, while the five-flow case shows very limited improvement in average goodput. Fig. 5.32 shows that Jain fairness is drastically improved, such that even though goodput did not greatly improve for the five-flow case, the log utility for that case is higher in Fig. 5.32(d).

Figures 5.33 and 5.34 include results from TCP-AP flows using 1-hop delay. The n-hop delay seemed to have the highest goodput for one-hop flows when $n = 1$ in the experiments described in Section 5.3. However, these results were also high in fairness and poor in goodput, though not as poor as the 4-hop delay flows. Jain fairness was unusually high for Tahoe in the two-flow case shown in Fig. 5.34(a), beating the 0.993 index that the one-hop delay achieved. The four-hop delay's lower rate, however, achieves a near-perfect Jain fairness index. For five flows, the Jain fairness is higher than Tahoe in either case, and the four-hop delay continues to provide better Jain fairness than the one-hop delay. Log utility in all cases is poorer than Tahoe due to diminished goodput.

We note here that we were unable to continue our experiments on parallel chains due to problems with the first-floor chain, and so we next consider our combined protocol on the short asymmetric chain. Results are seen in Fig. 5.35. Goodput here, as in the chain experiments, does not improve. Interestingly, fairness in this topology actually underperforms TCP Tahoe by a small margin. Diminished log utility is consistent with decreased goodput, but the Jain fairness result

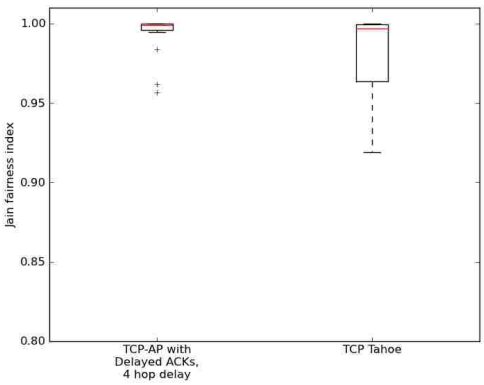


(a) 1 flow

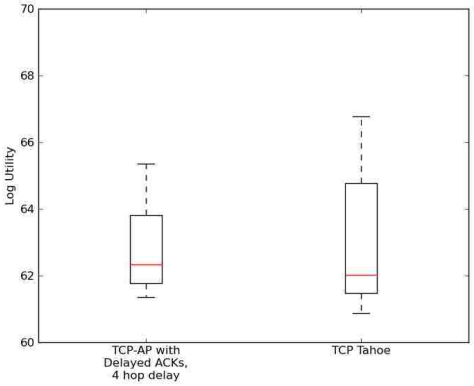
(b) 2 flows

(c) 5 flows

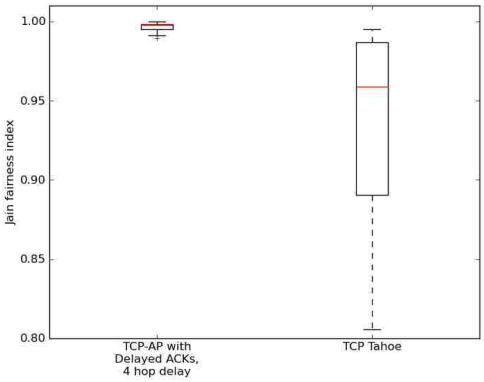
Figure 5.31: Goodput results for combined TCP-AP and Delayed ACKs on the 4-hop chain



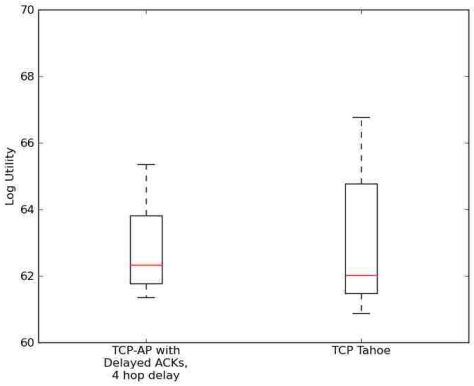
(a) Jain fairness - 2 flows



(b) Log utility - 2 flows

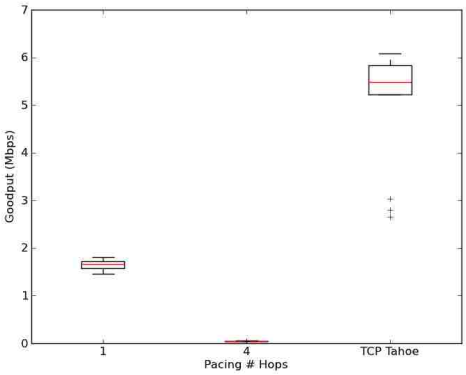


(c) Jain fairness - 5 flows

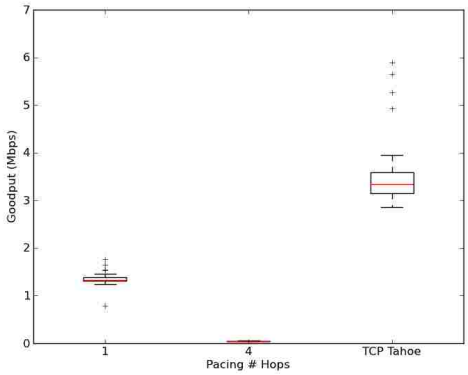


(d) Log utility - 5 flows

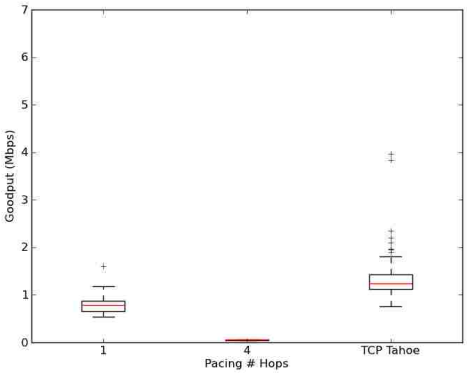
Figure 5.32: Fairness results for combined TCP-AP and Delayed ACKs on the 4-hop chain



(a) 1 flow

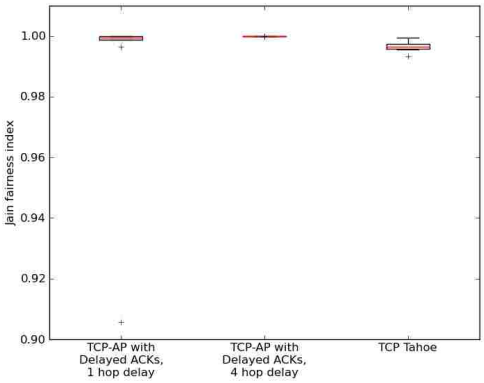


(b) 2 flows

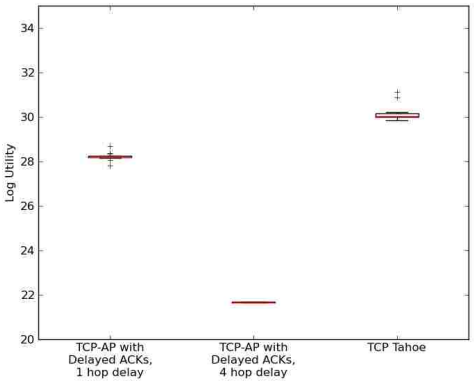


(c) 5 flows

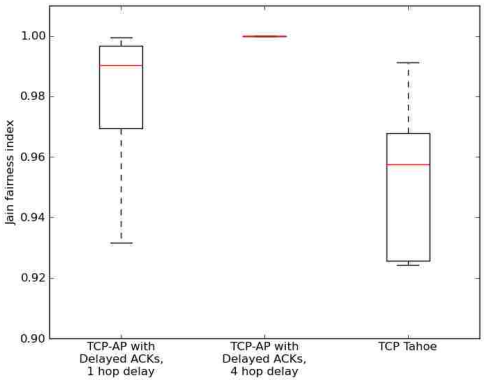
Figure 5.33: Goodput results for combined TCP-AP and Delayed ACKs on one hop



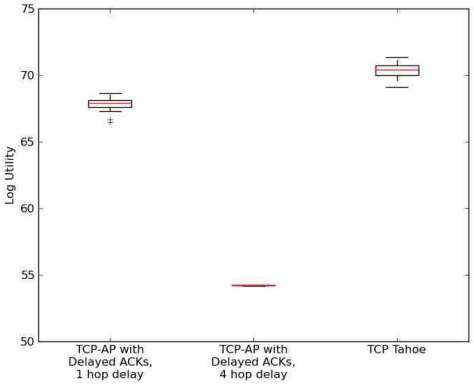
(a) Jain fairness - 2 flows



(b) Log utility - 2 flows

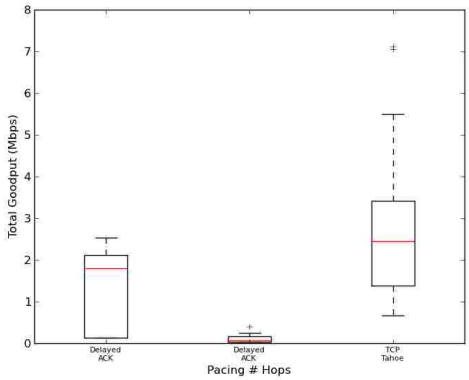


(c) Jain fairness - 5 flows

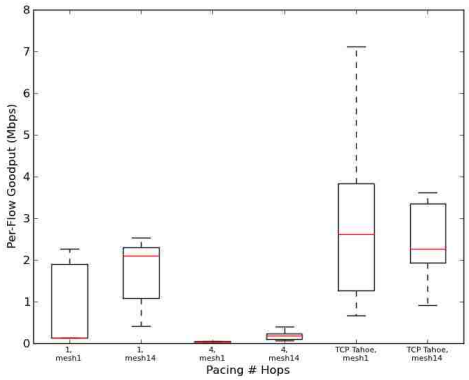


(d) Log utility - 5 flows

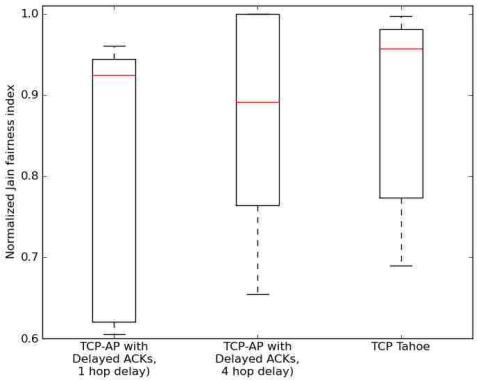
Figure 5.34: Fairness results for combined TCP-AP and Delayed ACKs on one hop



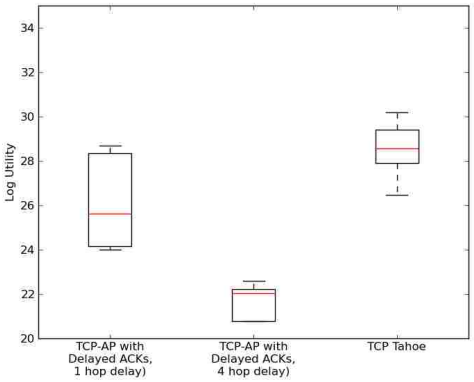
(a) Goodput



(b) Per-flow goodput



(c) Normalized Jain fairness



(d) Log utility fairness

Figure 5.35: Results for combined TCP-AP and Delayed ACKs on short asymmetric chains

seems at odds with the result for TCP-AP alone in Fig. 5.9(c). Note, however, that the normalized Jain fairness for $n = 4$ is far better than that for $n = 1$; since the optimal rates for each flow must now be different, it may be that unfairness is due to incorrect rate calculation. Another possibility is that the increased interference may cause ACKs to be dropped with sufficient frequency to overcome the fairness advantage of pacing; further investigation is needed to determine the cause of this anomalous unfairness.

To summarize, we conclude that the adaptive pacing approach used in TCP-AP [15] does not combine well with dynamically delayed ACKs to improve goodput in chain topologies or in the asymmetric chain topology. The n-hop delay metric yields poor goodput rates on these topologies. Combining with dynamically delayed ACKs as implemented in [9] yields no goodput benefit in our experiment; the advantage of delayed ACKs is nullified if the transmission rate is kept slow. Thus delayed ACKs without the FHD pacing metric outperform this combined approach by a large margin in terms of goodput and log utility. We do, however, continue to see an improvement in Jain fairness similar to what we see in our pacing-only experiments. This leads us to also conclude that further work is needed to determine a more appropriate pacing metric to combine with dynamically delayed ACKs, allowing us to see the goodput benefit of delayed ACKs while still rate-limiting flows in a fair manner.

5.7 Random Flows

5.7.1 Methodology

In order to approximate real traffic flowing over the wireless mesh network, we randomly select sources and sinks in the topology seen in Fig. 4.4. Routes through the topology are set up ahead of time using Djikstra's algorithm. We choose to test only the two most successful protocols, TCP-AP and TCP with delayed ACKs, along with our TCP Tahoe baseline. The experiments are repeated 10 times, with the experiment runs interleaved in order to reduce the effect of network conditions at different times of day. For each experiment run, flows are started simultaneously so that different numbers of flows approximate various levels of traffic intensity on the network.

There are some timing difficulties with the Python library used to signal the flows to start that causes some flows to fail, generally because the receiver thread starts after the sender thread.

# flows	% runs failed
32	21
16	12
8	26
4	10
1	10

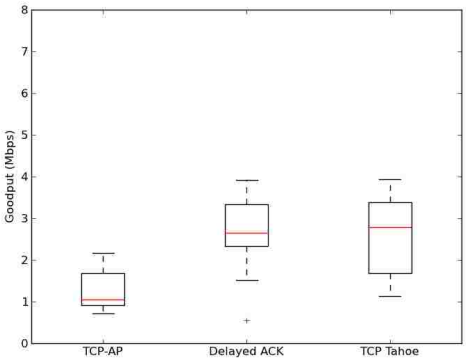
Table 5.2: Failure rates for the random flows experiments

Such flows must be discarded, and runs with higher numbers of simultaneous flows tend to encounter the problem more frequently. See Table 5.2 for the failure rates for each experiment. We run each experiment 10 times and average the results of all flows in order to eliminate the effect of these failures on our results.

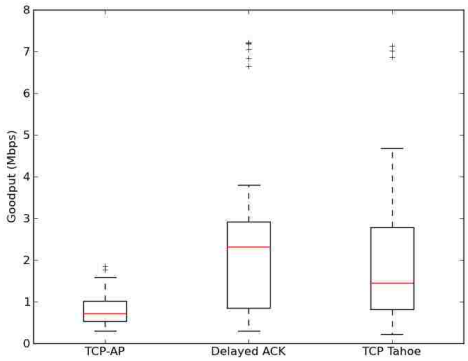
5.7.2 Results

The results are seen in Fig. 5.36. We do not analyze Jain fairness in this experiment because we do not know which flows are competing with some sort of bottleneck, nor is it worthwhile to determine the number of hops for each flow for normalization.

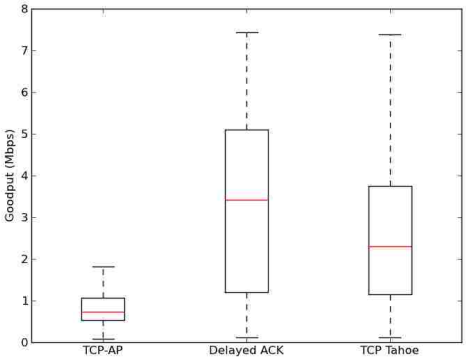
As our previous pacing experiments demonstrated, TCP-AP with a four-hop delay heuristic does a poor job of increasing goodput at every level of traffic intensity. Delayed ACKs also behave as expected with generally higher median goodputs and very high log utility when compared with the baseline.



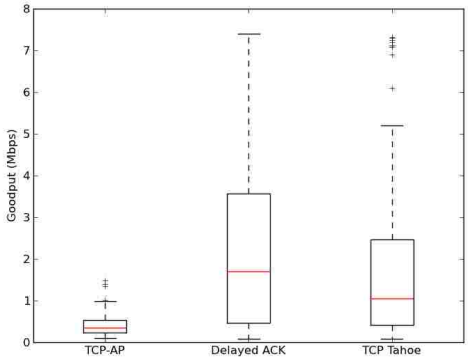
(a) One flow



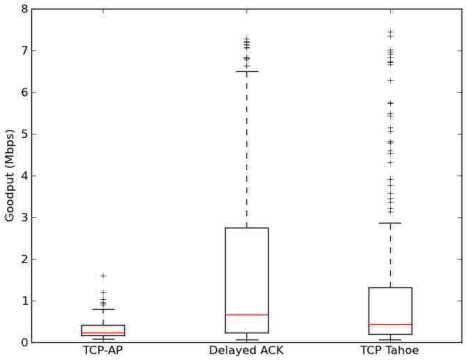
(b) Four flows



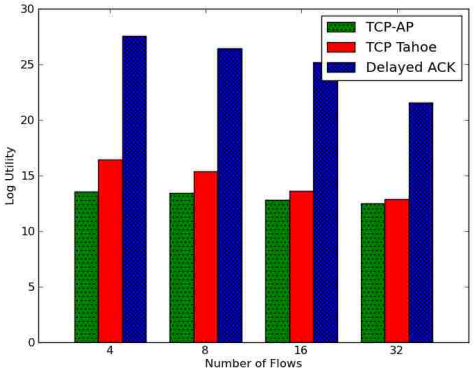
(c) Eight flows



(d) Sixteen flows



(e) Thirty-two flows



(f) Log Utilities

Figure 5.36: Results for TCP-AP and Delayed ACKs with random flows

Chapter 6

Conclusions and Future Work

TCP-AP [15] claims up to 84% increase in goodput over TCP NewReno, but we find that there is in fact a decrease in goodput over realistic path lengths. For long paths and a high number of competing flows, there may be a minor improvement, but certainly nothing near 84%. However, there is a definite fairness benefit gained by rate-limiting TCP, and it is possible that with the right rate, pacing may result in improved fairness with the same overall goodput as TCP. The recent work with adaptive pacing in [14] also requires evaluation.

TCP-FeW [27] reports a 90% increase in throughput over TCP NewReno. Conservative windows do not, however, perform better than TCP Tahoe; we conclude that this approach is not effective in statically-routed stationary wireless mesh networks. Note that because TCP-FeW [27] claims to improve TCP performance in the presence of mobility and dynamic routing, there is future work in testing this approach in both mobile and dynamically routed networks.

Altman and Jiménez [9] claim only a 20-40% increase in throughput over TCP, but this claim is more than borne out in our experiments; we find that dynamically delayed ACKs can increase goodput by up to 86%. Like TCP, dynamically delayed ACKs do not perform well in terms of Jain fairness, but all flows, including "starved" flows, tend to have increased goodput, yielding significant gains in log utility.

We conclude that conservative windows and n-hop delay are insufficient for improving performance in a static wireless mesh network, but there is potential for pacing to combine with dynamic delayed ACKs to improve TCP performance. It would be interesting future work to determine a method whereby TCP's window may be converted to a pacing rate and used to improve fairness among Delayed ACK flows.

We observe that of these three approaches, only Delayed ACKs [9] has been previously tested on an actual network. It is also the only approach whose throughput claims we are able to reproduce

and exceed on our network. This casts doubt on the frequent practice of evaluating transport protocols in simulation only; it is beyond the scope of this thesis to determine the limitations of simulator wireless models, but our results indicate that simulator throughput results on wireless mesh networks do not accurately reflect results measured in the real world.

References

[1] "Champaign-Urbana Community Wireless Network." [Online]. Available: http://www.cuwireless.net

[2] "Dolphin Stadium scores with world's largest wireless POS system powered by BelAir Networks." [Online]. Available: http://www.belairnetworks.com/news/press_releases/view.cfm/p_id/104

[3] "Freifunk." [Online]. Available: http://start.freifunk.net/

[4] "MIT Roofnet." [Online]. Available: http://pdos.csail.mit.edu/roofnet/doku.php

[5] "Seattle Wireless." [Online]. Available: http://www.seattlewireless.net/

[6] "SFLan." [Online]. Available: http://www.sflan.org/

[7] "Southampton Open Wireless Network." [Online]. Available: http://www.sown.org.uk/

[8] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Computer Networks and ISDN Systems*, vol. 47, pp. 445–487, March 2005. [Online]. Available: http://portal.acm.org/citation.cfm?id=1071644.1071646

[9] E. Altman and T. Jiménez, "Novel delayed ACK techniques for improving TCP performance in multihop wireless networks," *Personal Wireless Communications*, pp. 237–250, September 2003.

[10] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 756–769, December 1997.

[11] B. Bensaou, Y. Wang, and C. C. Ko, "Fair medium access in 802.11 based wireless ad-hoc networks," in *1st Annual Workshop on Mobile and Ad Hoc Networking and Computing*, 2000, pp. 99 –106.

[12] R. Bruno, M. Conti, and E. Gregori, "Mesh networks: commodity multihop ad hoc networks," *IEEE Communications Magazine*, vol. 43, no. 3, pp. 123–131, March 2005.

[13] R. Buck, "WiFu transport: a user-level protocol framework," Master Thesis, Dept. of Computer Science, Brigham Young University, June 2012.

[14] S. M. ElRakabawy and C. Lindemann, "A practical adaptive pacing scheme for TCP in multihop wireless networks," *IEEE/ACM Transactions on Networking*, vol. 19, pp. 975–988, August 2011. [Online]. Available: http://dx.doi.org/10.1109/TNET.2010.2095038

[15] S. ElRakabawy, A. Klemm, and C. Lindemann, "TCP with adaptive pacing for multihop wireless networks," in *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. ACM, 2005, pp. 288–299.

[16] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno and SACK TCP," *ACM SIGCOMM Computer Communication Review*, vol. 26, pp. 5–21, July 1996. [Online]. Available: http://doi.acm.org/10.1145/235160.235162

[17] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP throughput and loss," in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3. IEEE, 2003, pp. 1744–1753.

[18] V. Gambiroza, B. Sadeghi, and E. W. Knightly, "End-to-end performance and fairness in multihop wireless backhaul networks," in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*. New York, NY, USA: ACM, 2004, pp. 287–301. [Online]. Available: http://doi.acm.org/10.1145/1023720.1023749

[19] M. Gerla, K. Tang, and R. Bagrodia, "TCP performance in wireless multi-hop networks," in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*. IEEE, 1999, p. 41.

[20] O. Gurewitz, V. Mancuso, J. Shi, and E. W. Knightly, "Measurement and modeling of the origins of starvation of congestion-controlled flows in wireless mesh networks," *IEEE/ACM Transactions on Networking*, vol. 17, pp. 1832–1845, December 2009. [Online]. Available: http://dx.doi.org/10.1109/TNET.2009.2019643

[21] J. Heidemann, N. Bulusu, J. Elson, C. Intanagonwiwat, K. Lan, Y. Xu, W. Ye, D. Estrin, and R. Govindan, "Effects of detail in wireless network simulation," in *Proceedings of the SCS Multiconference on Distributed Simulation*, January 2001, pp. 3–11. [Online]. Available: http://www.isi.edu/~johnh/PAPERS/Heidemann00d.pdf

[22] G. Holland and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," *Wireless Networks*, vol. 8, no. 2/3, pp. 275–288, March-May 2002.

[23] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM Computer Communication Review*, vol. 18, pp. 314–329, August 1988. [Online]. Available: http://doi.acm.org/10.1145/52325.52356

[24] R. Jain, D. ming Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," Digital Equipment Corporation, Tech. Rep., September 1984.

[25] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott, "Experimental evaluation of wireless simulation assumptions," in *Proceedings of the International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*, 2004, pp. 78–82.

[26] J. Liu and S. Singh, "ATCP: TCP for mobile ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 7, pp. 1300–1315, July 2001.

[27] K. Nahm, A. Helmy, and C. Jay Kuo, "TCP over multihop 802.11 networks: issues and performance enhancement," in *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing.* ACM, 2005, pp. 277–287.

[28] C. Parsa and J. Garcia-Luna-Aceves, "Improving TCP performance over wireless networks at the link layer," *Mobile Networks and Applications*, vol. 5, no. 1, pp. 57–71, March 2000.

[29] K. M. Reineck, "Evaluation and comparison of network simulation tools," Master Thesis, Dept. of Computer Science, University of Applied Sciences Bonn-Rhein-Sieg, August 2008. [Online]. Available: http://projektiv.net/karsten/master-thesis/Reineck_-_Evaluation_and_Comparison_of_Network_Simulation_Tools.pdf

[30] K. Sundaresan, V. Anantharaman, H. Hsieh, and R. Sivakumar, "ATP: A reliable transport protocol for ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 4, no. 6, pp. 588–603, 2005.

[31] F. Wang and Y. Zhang, "Improving TCP performance over mobile ad-hoc networks with out-of-order detection and response," in *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing.* ACM, 2002, pp. 217–225.

[32] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?" *IEEE Communications Magazine*, vol. 39, no. 6, pp. 130 –137, June 2001.

[33] X. Yu, "Improving TCP performance over mobile ad hoc networks by exploiting cross-layer information awareness," in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking.* ACM, 2004, pp. 231–244.