

Limitations of TCP-ELFN for Ad hoc Networks

Jeffrey P. Monks, Prasun Sinha and Vaduvur Bharghavan
Coordinated Science Laboratory, University of Illinois at Urbana-Champaign
1308 W. Main Street, Urbana, IL 61801
{jmonks, prasun, bharghav}@crhc.uiuc.edu

Abstract

Research on congestion control in ad hoc networks has mostly been limited to enhancing TCP based mechanisms [1, 2, 3, 4, 5]. Explicit Link Failure Notification (ELFN [1, 2]), a solution for ad hoc TCP, has been shown to outperform TCP in mobile scenarios. Studies of TCP+ELFN on static networks show throughput degradation of about 5% in several cases over TCP, however, the improvements in dynamic cases can be as large as 7 times, as shown in presented simulations. We further describe and study various limitations of TCP based approaches in general, and TCP+ELFN in particular, for congestion control on ad hoc networks. Based on our studies on queue sizes and the progression of TCP sequence numbers for various scenarios, we observe that for congestion control in ad hoc networks, hop-by-hop rate control based mechanisms along with ELFN will be better suited than TCP based mechanisms.

Keywords: TCP, Ad hoc TCP, ECP-ELFN

1 Introduction

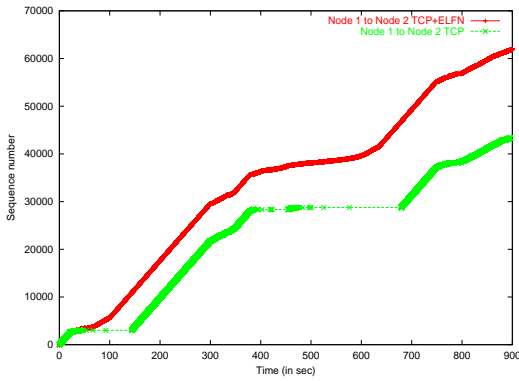
The rapid increase in the number of PDA (Personal Digital Assistants) devices, palmtops and compact laptops has made ad hoc networks [6] a promising technology of the near future. The shared nature of the channel and mobility of nodes in such networks cause heavy contention and large variations in the available bandwidth. These poor properties of the channel in ad hoc networks and the increasing number of multimedia applications, has instigated research on the transport layer.

Several mechanisms, mostly enhancements to TCP, have been proposed for ad hoc networks [1, 2, 3, 4, 5]. The Explicit Link Failure Notification (ELFN[1, 5]) mechanism has been simulated in conjunction with the TCP protocol, and shown to outperform TCP in dynamic scenarios. However, we look at both static and dynamic scenarios with TCP+ELFN in this paper. TCP based solutions suffer from various limitations in such environments, due to some inherent properties of TCP as mentioned below.

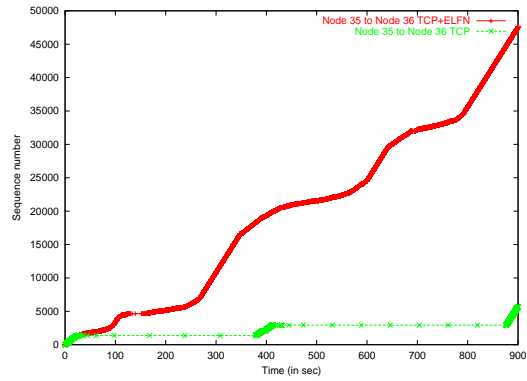
- *TCP creates losses until it starts observing losses:* TCP's congestion control mechanism is based on reacting to observed losses and probing for available bandwidth by progressively increasing the congestion window. This nature of TCP results in high network buffer utilization which causes large RTT and therefore, large RTOs for TCP flows. Large RTOs result in long timeouts, thus causing reduced TCP throughput.
- *TCP takes several RTTs to recover after timeouts:* TCP throughput is affected due to timeouts which reduce the congestion window size to 1. After this abrupt reduction in sending rate, it takes several RTTs before the TCP sender reaches the congestion avoidance phase.
- *Route disruptions can cause cwnd worth of data losses:* TCP can have a *cwnd* worth of data outstanding in the network. As link failures are inherent in ad hoc networks, the data in the network buffers can be lost due to link failures and subsequent failures in salvaging packets. Just using low congestion window values is not sufficient as it may not guarantee efficient channel utilization. So the congestion control mechanism should require small network buffers.

These properties limit the performance of TCP based solutions such as TCP+ELFN. This affect is analyzed by studying the progression of sequence numbers and network buffers for various scenarios. Based on these studies we conclude that hop-by-hop rate control along with the mechanisms of ELFN, will perform better than TCP based solutions.

The rest of the paper is organized as follows: Section 2 presents the simulation environment used for all the studies presented in the paper. The ELFN algorithm and simulations on static and dynamic networks are presented in Section 3. We analyze the buffers and study the fairness properties of multiple flows in static and dynamic networks in Section 4. Finally, the paper is concluded in Section 5.



(a)



(b)

Figure 1: Comparison of a TCP+ELFN and a TCP flow in a dynamic scenario for two different pair of end hosts. The scenario is 50 nodes in a 1500m×300m area with maximum speed of 20m/s and zero pause time.

2 Simulation Environment

For all our simulation studies, we have used the *ns2* [7] simulator and its ad hoc extensions provided by the Monarch [8] research group at CMU. Like previous work on ELFN [1, 2, 5], we have also used the DSR [6] protocol for routing in our simulations.

Our tests are on mobile and static scenarios. The packet size used is 1460 bytes of data. The channel bandwidth is 2Mbps and the transmission range of the nodes is 250m. The random *waypoint model* [6] was used for mobility which has two key parameters namely the maximum speed and the pause time. For all our mobility experiments we have used a network of 50 nodes on a 1500m×300m area with a pause time of 0 and a maximum speed of 20 m/s. Note that this particular network is never disconnected and hence, there is always a route between any pair of nodes.

3 Explicit Link Failure Notification (ELFN)

This section first gives an overview of the ELFN technique, followed by some simulation studies of TCP+ELFN on static and dynamic networks.

Several researchers [1, 5] have proposed TCP enhancements based on freezing the network state at the TCP source using explicit messages in the event of route failures. The explicit link failure notification (ELFN) technique described in [1] is shown to outperform TCP for various mobile scenarios. The following mechanisms proposed in [1] were introduced in the *ns2* simulator to implement ELFN:

- *Disable response from caches:* In DSR, a node responds to the route request if it is the destination or if it has a cached route to the destination. Holland et. al. have shown in [2] that TCP performance is improved by disabling cached routes,

as caching helps in propagation of stale route information. So we disabled responding to route requests based on cached routes.

- *Freezing the sender state:* DSR sends a RERR (Route Error) message to the sender on a link failure. Link failures are detected either by the ARP or the MAC layer upon failure to get a ARP reply or failure to deliver a packet, respectively. When the TCP sender receives a route failure indication (through RERR message), it freezes its congestion window and timers, and enters the *frozen state*.
- *Probing in the frozen state:* In the *frozen state*, the sender probes every 2 seconds by sending a probe packet, to check the validity of the route. The unacknowledged packet with the lowest sequence number is used as the probe packet.
- *Leaving frozen state:* On reception of an ACK for a probe packet (the probe packet and the corresponding ACK are specially marked), the TCP sender leaves the frozen state, starts running the timers which were frozen, and resumes normal operation.

These ELFN mechanisms targeted towards handling route failures, improve the performance of TCP in dynamic scenarios. For static scenarios, the freezing due to RERR messages may reduce the performance, and we investigate this in the next section where we study the performance of ELFN in dynamic as well as static scenarios.

3.1 ELFN+TCP in dynamic and static networks

To illustrate the advantage of ELFN+TCP over TCP for mobile cases, we present results of a TCP+ELFN flow in the 50 node scenario. To contrast with the

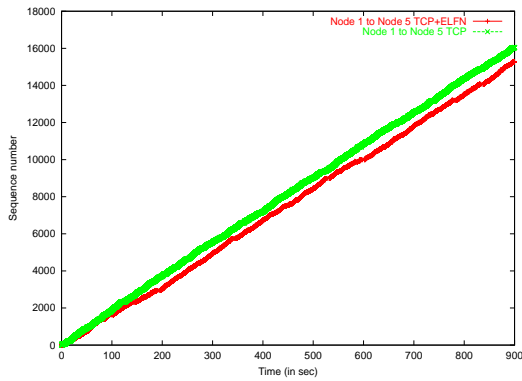


Figure 2: Static network with 1 flow on a 5 hop chain: Comparison of TCP+ELFN and TCP flows.

performance of TCP, we repeat the test with a TCP flow, instead of a TCP+ELFN flow and present the result in Figure 1(a). Figure 1(b) presents the same experiment repeated for a different set of end hosts.

- *Figure 1(a)*: The TCP+ELFN flow shows a throughput improvement of 43%. In this case, the source froze for 119 times in the 900s simulation.
- *Figure 1(b)*: For this source-destination pair, the TCP+ELFN flow is able to send more than 7 times as much data as the TCP flow. Here, the source froze 5 times during the 900 seconds.

In both the examples (Figures 1(a) and 1(b)), we observe that timeouts can lead to stalling of the TCP flows resulting in low throughput. As mentioned earlier, *TCP does not have any mechanism to distinguish packet losses due to route failures from losses due to congestion*. Link breakages, which are common in mobile networks, cause the source to cut down its window size, due to observed packet losses. This leads to timeouts, and repeated timeouts result in reduced throughput as shown in the figures. TCP+ELFN does not allow the congestion window, timer or any other TCP state to be affected during the search for a new route. Thus, ELFN tries to mask the effect of losses due to link failures in mobile scenarios.

TCP+ELFN was designed mainly for dynamic networks to alleviate the problem of TCP throughput degradation due to link failures. Earlier studies on ELFN [1, 2, 5] do not present performance results on static networks or with multiple simultaneous flows (analyzed in Section 4). The behavior of TCP+ELFN on ad hoc static networks is very different from that on wired networks, due to the following three main properties specific to the channel characteristics of ad hoc networks. First, the channel is shared, unlike in wired networks and so the available bandwidth is greatly reduced compared to the raw

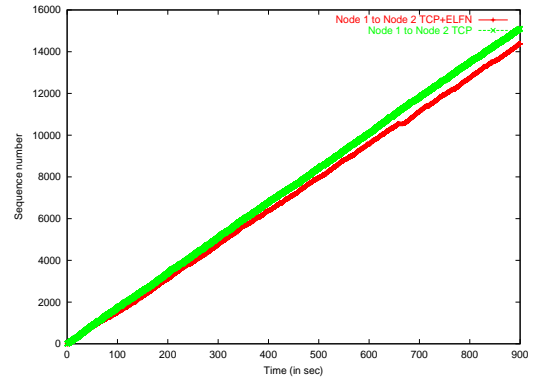


Figure 3: Static network with 1 flow on a 50 node network in a 1500m×300m area: Comparison of TCP+ELFN and TCP flows.

bandwidth; second, packets from the same flow contend with each other; and third, every node gets a share of the channel based on the fairness properties of the MAC layer. In the case of static ad hoc networks, if the flows create enough channel contention to cause frequent route error messages, then freezing the network state for the RERR messages could lead to reduced throughput. To study this effect, we present here some performance tests of TCP+ELFN on static networks. The throughput of TCP+ELFN was seen to suffer due to false link failure detections. We present two experiments illustrating the same for two different static networks:

- *Chain of 5 nodes*: The network is a chain of 5 nodes, each separated by the transmission range (250 m). A TCP flow was then run between the chain's end nodes. We then compared the throughput results with a TCP+ELFN flow. The two graphs are shown in Figure 2. In the 900 seconds run of the simulation, the TCP+ELFN sender froze the TCP state 374 times. This resulted in 5% lower throughput of the TCP+ELFN flow compared to a TCP flow.
- *50 nodes in a 1500m×300m network*:

In this static network we picked up a source and a destination node, and ran a TCP flow between the two nodes for 900 seconds. We repeated the same with a TCP+ELFN flow. The two graphs are shown in Figure 3. The TCP+ELFN sender froze for 455 times in the 900 seconds run. Thus, route changes due to contention, again lead to a throughput of 5% lower for TCP+ELFN in comparison to TCP.

This shows that in static networks, even a single flow is capable of creating enough contention to cause performance degradation due to ELFN's strategy of freezing TCP state on reception of a RERR messages. Note that the observed TCP throughput in

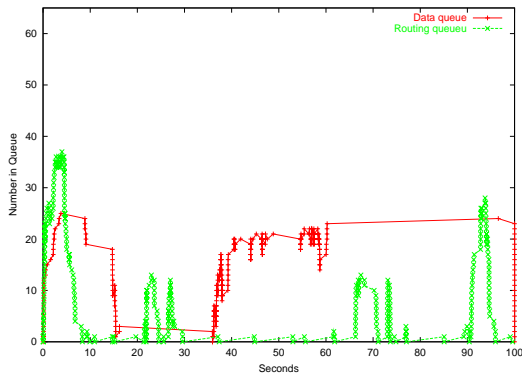


Figure 4: Routing and normal queues during a simulation of a 1500x300 meters static network with 50 nodes and 10 flows

both the experiments is roughly 0.2 Mbps, which is only 10% of the raw capacity. This is a result of various mechanisms such as the contention caused among data packets, the contention caused by ACK packets, the MAC contention algorithm, and TCP’s saw-tooth behavior in the congestion avoidance phase.

4 TCP based mechanisms: Study of Fairness Issues and Queue Sizes

In this section, we study the behavior of TCP+ELFN from the user’s perspective and from the network’s perspective. End-to-end behavior (user’s perspective) is studied by analyzing the sequence number progression for multiple flows. Study of network buffers gives an understanding of the protocol from the network’s perspective. We bring out several queuing and fairness related issues for multiple flows in static as well as dynamic networks. Based on these studies we propose hop-by-hop rate control as a method for improving performance of TCP.

In the remaining section, we first present a study of the network buffers, followed by some fairness issues observed in these simulations, and finally concluding with a comparison of end-to-end and hop-by-hop rate control methods. For all simulations presented in this section, 10 TCP flows were simulated.

4.1 Network Buffers

As discussed in Section 1, the amount of buffering in the network affects the RTT and RTO estimates, which in turn affects the period of timeouts and thus, the net throughput of TCP connections. Study of network buffers also helps in understanding buffer overflows, if they are occurring. This can lead to improved protocol designs or may justify the need

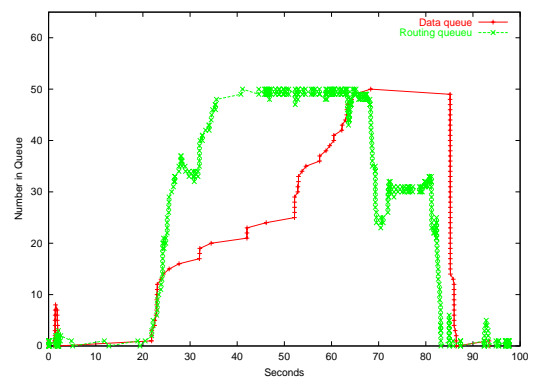


Figure 5: Routing and normal queues during a simulation of a 1500x300 meters mobile network with 50 nodes and 10 flows

for provisioning more buffer space. Here, we study network buffers for static and dynamic scenarios.

In the *ns2* simulator, separate prioritized MAC queues are maintained for various packet types with highest to lowest priority queues being routing, real-time, low-delay, and normal packets. A packet is only selected from a queue when its higher priority queues are empty. For our simulations, the real-time queue was not used. The routing packets (DSR in our case) are stored in the routing queue, ACK packets are considered low-delay, and ARP and TCP data packets are classified as normal packets. The graphs presented only show the routing and the normal queues, where the routing queue has higher precedence. The maximum queue size of all queues is 50.

Figure 4 shows the routing and normal queue sizes of the intermediate node with highest average normal queue size for a static network. In this figure, the queue sizes never exceed the maximum capacity because the routing packets arrive in short, infrequent bursts. Note that even for static networks, routing packets are generated during data transmission as a result of extensive packet delays (for reasons such as contention caused by the same flow) causing nodes to assume link breakages. The static case shows that the network can handle the traffic level without congestion or packet drops. A similar graph is presented for the dynamic scenario in Figure 5. Despite the fact that the graph corresponding to this ad hoc network is always connected, the routing and normal queues are filled for a considerable period of the simulation because of congestion and changes in node connectivity. Note that the number of packets in the normal queue continues to grow as long as routing packets are present and are not dequeued until no routing packets remain. The fact that the static network can handle these 10 flows, indicates that it is rather the mobility that causes link breakages and significant routing overhead resulting in congestion.

High mobility of nodes causes frequent route disruptions, resulting in large number of routing packets, which force the TCP packets to reside in the network buffers for significant periods of time, causing many timeouts. Therefore, queue management is more critical for mobile networks, where data packets can be significantly delayed or dropped due to changes in network configuration. To reduce incidence of queue overflows, congestion control mechanisms with high efficiency and low buffer utilization are needed. Unlike TCP, mechanisms based on rate control do not burst out packets and keep the buffer utilization low. Such mechanisms need further investigation in the realm of ad hoc networks for congestion control.

4.2 Fairness Issues

In wireless networks, users frequently receive an unfair allocation of system resources due to one node taking control of the channel or one flow reducing its congestion window, thus allowing another maintain a larger window and send more packets. This problem was investigated for TCP [9] and therefore is also present even with TCP+ELFN. Figure 7 shows that in a mobile network with multiple flows the throughput can be significantly different for competing flows. This is particularly evident when comparing short range flows (those requiring only a few hops) to longer range flows (those requiring a larger number of hops). The short range flows have fewer timeouts and a larger congestion window causing them to contend more aggressively. In Figure 6, the sequence number progression is shown for the static network. The figure only shows the two flows receiving the highest and two receiving the lowest service from the network. Here we see similarly that there is a great difference in packet deliveries between short range and long range flows. In this figure, Flow 1 (requiring one hop) sends significantly more packets than Flow 4 (requiring five hops) as a result of short range flows contending more aggressively, which causes longer range flows to back-off and further reduce their contention window. In the static network both flows 3 and 4 require only one hop between source and destination, however, their performance is substantially below that of flow 1 because they reside in an area of the network where a greater number of nodes are competing for the channel. Noting that the graph corresponding to Figure 7 is always connected, we see that Flow 1 (requiring one hop on average) sends considerably more packets than Flow 4 (requiring four hops on average). Even if we take into account the fact that multiple flows are contending, we have observed that both ELFN and plain TCP result in an unfair network resource distribution due to the congestion at intermediate hops. Notice that in the mobile case, flows 2, 3, and 4 maintain the same level for some periods during the

simulation while ELFN freezes the TCP timers and sends probe packets in search of a valid route. In all these studies we observe that large bursts from one flow can cause other flows to assume congestion and backoff. Mechanisms based on rate control attempt at avoiding bursts of data, and therefore, would be more suited for ad hoc environments.

4.3 Rate control methods

There are two choices for rate control, end-to-end or hop-by-hop rate control. Hop-by-hop rate control involves each node individually controlling the outgoing rates of all flows passing through it such that the outgoing MAC or downstream nodes are not overwhelmed. End-to-end rate control mechanisms could be dependent or independent of network feedback. However, end-to-end rate control has some disadvantages over hop-by-hop rate control, which are explained below.

- *No large buffer buildups:* With end-to-end rate control, when intermediate nodes receive large bursts of packets from a particular flow, later packets received from other flows must wait for all packets from the preceding flow to contend for transmission. Thus, end-to-end mechanisms may lead to large buffer buildups, whereas hop-by-hop methods never result in large buffer buildups as the per hop rate control mechanism prevents the buffers from getting large.
- *Fairness among long and short flows:* In case of end-to-end mechanisms, the longer flows reduce the sending rate when congestion is observed in *any* of the links on the path. However, in case of hop-by-hop rate control, congestion causes only the immediate hop to reduce the sending rate and only when it persists, is the effect carried over to the sender. Also, the RTT for long flows is larger than short flows, and thus results in slower rate updates for end-to-end mechanisms based on RTT. Hop-by-hop mechanisms are RTT independent and do not suffer from unfairness caused due to difference in RTT between flows.
- *More responsive:* Hop by hop rate control is more responsive to changes in connectivity, congestion, or number of contending nodes than end-to-end methods. End-to-end rate control requires on the order of a few round trip times to adjust the rate, while hop-by-hop rate control can adjust the outgoing rate of a particular node on the order of a single packet transmission time.
- *Low buffer requirement on stale routes:* Hop-by-hop transmission can reduce the number of packets stored at a disconnected node (node whose

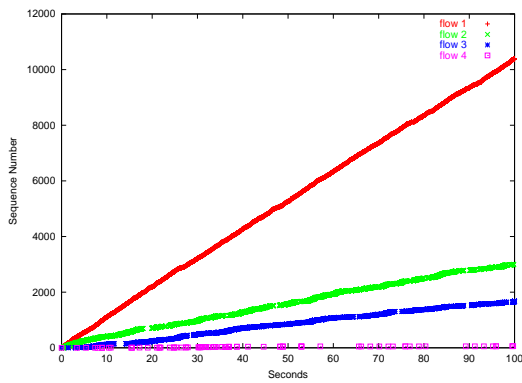


Figure 6: Packet sequence nos. for a simulation of a $1500m \times 300m$ static network with 50 nodes and 10 flows (only 4 shown)

next hop is broken) in a stale route by requesting that upstream nodes reduce their sending rate or stop sending all together immediately after the breakage is detected.

Hop-by-hop rate control has two main drawbacks. First, the amount of state that needs to be maintained at every node is of the order of the number of flows passing through the node. Second, the packets which are not held by the network, in case of hop-by-hop mechanism, has to be held at the sender. However, from the perspective of performance, and the network buffer requirement, the advantages of hop-by-hop mechanisms outweigh its drawbacks.

5 Conclusions

ELFN [1, 2] is a mechanism for improving the performance of TCP on ad hoc networks. It has however been studied only for dynamic networks and only in conjunction with TCP. We present results on static as well as dynamic networks and show that the throughput in static networks can be around 5% lower than in the case of TCP and the throughput improvement in case of dynamic networks can be very high (one of our examples shows about 7 times improvement). We present studies of the network buffers and the progression of sequence numbers of TCP+ELFN, to demonstrate limitations of TCP based congestion control mechanisms for ad hoc networks. These studies lead us to conclude that hop-by-hop rate control mechanisms along with ELFN is better suited for ad hoc networks rather than TCP based mechanisms, such as TCP+ELFN.

References

[1] G. Holland and N. H. Vaidya, “Analysis of TCP Performance over Mobile Ad Hoc Networks,” in

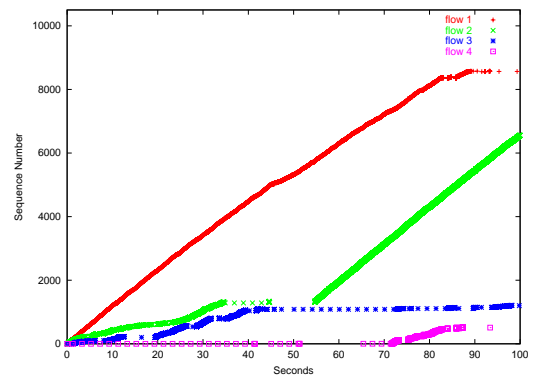


Figure 7: Packet sequence nos. for a simulation of a $1500m \times 300m$ mobile network with 50 nodes and 10 flows (only 4 shown)

Proc. IEEE MOBICOM, Seattle, Aug. 1999.

- [2] G. Holland and N. H. Vaidya, “Impact of Routing and Link Layers on TCP Performance in Mobile Ad Hoc Networks,” in *Proc. WCNC*, New Orleans, Sept. 1999.
- [3] M. Gerla, K. Tang, and R. Bagrodia, “TCP performance in wireless multi-hop networks,” in *Proceedings of Second IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, New Orleans, LA, 1999, pp. 41–50.
- [4] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, “A Feedback based Scheme for Improving TCP Performance in Ad hoc Wireless Networks,” in *Proc. ICDCS*, Amsterdam, May 1998.
- [5] B. Bakshi, P. Krishna, N. H. Vaidya, and D. K. Pradhan, “Improving Performance of TCP over Wireless Networks,” in *Proc. ICDCS*, Baltimore, may 1997.
- [6] J. Broch, D. B. Johnson, and D. A. Maltz, “The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks,” Internet Draft draft-ietf-manet-dsr-03.txt, Oct. 1999.
- [7] K. Fall and K. Vardhan, “*ns* notes and documentation,” available from <http://www-mash.cs.berkeley.edu/ns/>, 1999.
- [8] “Ad-hoc extensions to *ns*,” available from <http://monarch.cs.cmu.edu/>, 1999.
- [9] T. R. Henderson, E. Sahouria, S. McCanne, and R. H. Katz, “On improving the fairness of TCP congestion avoidance,” in *Proceedings of Global Telecommunications Conference (GLOBECOM)*, 1998, vol. 1, pp. 539–544.