TCP Performance in Flow-Based Mix Networks: Modeling and Analysis

Xinwen Fu, *Member, IEEE,* Wei Yu, Shu Jian, Steve Graham, and Yong Guan

***Abstract*—Anonymity technologies such as mix networks have gained increasing attention as a way to provide communication privacy. Mix networks were developed for message-based applications such as email, but researchers have adapted mix techniques to low-latency, flow-based applications such as anonymous web browsing. Although a significant effort has been directed at discovering attacks against anonymity networks and developing countermeasures to those attacks, there is little systematic analysis of the quality of service (QoS) for such security and privacy systems. In this paper, we systematically address TCP performance issues of flow-based mix networks. A mix's batching and reordering schemes can dramatically reduce TCP throughput due to out-of-order packet delivery. We developed a theoretical model to analyze such impact and present formulae for approximate TCP throughput in mix networks. To improve TCP performance, we examined the approach of increasing TCP's duplicate threshold parameter and derived formulae for the performance gains. Our proposed approaches will not degrade the system anonymity degree since they don't change the underlying anonymity mechanism. Our data matched our theoretical analysis well. Our developed theoretical model can guide deployment of batching and reordering schemes in flow-based mix networks, and can also be used to investigate a broad range of reordering schemes.**

***Index Terms*—Anonymity, Mix Networks, TCP, Congestion Control, Modeling and Analysis**

I. Introduction

Concerns about privacy and security have received greater attention with the rapid growth and public acceptance of the Internet. *Anonymity* has become a necessary and legitimate aim in many application areas, including anonymous web browsing and file sharing. In these scenarios, encryption alone cannot maintain the anonymity required by participants [1], [2], [3]. Since Chaum [4] pioneered the basic idea of anonymous communication system, referred to as *mixes*, researchers have developed various anonymity systems for different applications. Mix techniques can be used for either message-based (high-latency) or flow-based (low-latency) anonymity applications. Anonymous email is a typical message-based anonymity application, which has been thoroughly discussed [5]. However, flow-based mix networks, which provide services such as anonymous web browsing and file sharing, have different performance characteristics and requirements.

There are two requirements for a successful anonymous communication system: the degree of anonymity the system can achieve (anonymity degree) [6], [7] and the quality of service (QoS). Although a significant effort has been directed at discovering attacks against anonymity networks and developing countermeasures to those attacks, there is little systematic QoS analysis for such security and privacy systems.

This paper focuses on theoretical analysis and simulation study of mix network TCP performance for flow-based anonymity applications. In order to thwart a variety of traffic analysis attacks from degrading email anonymity, researchers have designed batching and reordering techniques for mixes to reduce timing correlation between packets entering a mix and those leaving from the mix [8]. *Intuitively, attacks against message-based anonymity applications can also be used against flow-based anonymity applications*. For example, flow-based mix networks can be attacked using packet (message) timing watermarks [9], [10]. Therefore, *batching and reordering defensive schemes should be considered for flow-based anonymity applications*. Anonymity analysis has been conducted on flow-based anonymous communication systems with batching and reordering applied [11], [12], [13]. However, it is worthwhile to study the performance of such schemes, in the context of low-latency applications. Danezis [14] utilized a Poisson model of traffic and conducted traffic analysis of continuous-time mixes (refer to Table I) for flow-based mix networks. However, a Poisson approach is limited as a model for TCP performance.

In this paper, we quantify the performance impact on flow-based applications when batching and reordering schemes are used in mix networks to improve the anonymity. This analysis formally explains the limited use of otherwise successful anonymity networks such as Tor [15] for flow-based applications. While generally poorer performance seems intuitive, our analysis demonstrates the counter-intuitive result that simply increasing link speed does not resolve the problem at all. Our major contributions are summarized as follows.

1. We systematically analyze TCP throughput in a mix network which uses batching and reordering schemes. Through theoretical analysis and simulations, we demonstrate that the out-of-order delivery introduced by batching and reordering schemes causes TCP to suffer a *bounded maximum congestion window* and dramatically degrades its throughput. We found that the throughput degradation does not improve even if the underlying link bandwidth is increased to infinity.

2. We analyze improving TCP throughput by varying TCP

Xinwen Fu is with the Department of Computer Science, University of Massachusetts Lowell, One University Avenue, Lowell, MA 01854 (Email:xinwenfu@cs.uml.edu).

Wei Yu and Shu Jiang are with the Department of Computer Science, Texas A&M University, 301 Harvey R. Bright Bldg, College Station, TX 77843 (Email:{weiyu, jiangs}@cs.tamu.edu).

Steve Graham is the College of Business and Information Systems, Dakota State University, 820 N. Washington Ave. Madison, SD 57042 (Email:skg@dsu.edu).

Yong Guan is with Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 (Email: yguan@iastate.edu).

parameters in a mix network using batching and reordering schemes. We develop analytical results for the TCP throughput gain on increasing TCP's duplicate threshold, $dupthresh$. Our simulations demonstrate the feasibility of this scheme. We also address limitations of other existing approaches in Sections IV-B and V-D. We emphasize that *the proposed approaches will not degrade the anonymity degree of systems since they don't change the underlying anonymity mechanism*.

3. Our analysis can guide the deployment of batching and reordering schemes in flow-based mix networks. Our theoretical analysis of TCP performance can be applied to a broad range of random reordering schemes, and our simulation results support the analytical model.

Many researchers have investigated adjusting $dupthresh$ for addressing TCP performance issues caused by out-of-order delivery in other contexts [16], [17], [18]. Although there has been work on the impact of out-of-order packet delivery on TCP QoS, to the best of our knowledge, we are the first to develop *theoretical models* for analyzing the impact of the out-of-order delivery caused by a mix network on TCP performance and present formulae for approximate TCP throughput for mix networks. We also conducted both thorough theoretical analysis and simulations of the impact of increasing $dupthresh$ on TCP throughput and presented approximate formulae, while others researching TCP throughput relied on *simulations*.

The remainder of this paper is organized as follows: We introduce mix techniques in Section II. In Section III, we analyze TCP performance degradation in flow-based mix networks using batching and reordering schemes. In Section IV, we analyze TCP performance gains from increasing $dupthresh$. In Section V, we use ns-2 simulations to corroborate our analytical results. We review related work in Section VI. We summarize the paper in Section VII.

II. Mix Techniques

A traditional mix is a relay server for anonymous email communication. It has a public key which senders use to encrypt messages. A mix operates as follows: (1) the sender attaches the receiver address to the message and encrypts the entire package by using the mix's public key; (2) the mix collects a batch of messages (from different senders) and decrypts them to obtain the receivers' addresses; (3) finally the mix sends decrypted messages out in a rearranged order to the receivers. Batching and reordering are necessary techniques for a mix to prevent traffic analysis attacks, which may correlate input messages and output messages by their timing.

A mix network consisting of multiple mix servers can provide enhanced anonymity. In a mix network, senders route their messages through a series of mixes. Therefore, even if an adversary compromises one mix and discovers the correlation between its input and output messages, other mixes along the path can still provide the necessary anonymity. Figure 1 illustrates the route selection for one message. A sender can choose different routes for each message or use one route for all her messages through the mix network [5], [19], [20].

Message-based mix networks have been extended to flow-based networks for low-latency applications such as anonymous FTP, Web browsing, video and audio transmission and many others [15]. In the context of an IP network, the relay servers in Figure 1 form an overlay network and forward packets instead of messages.

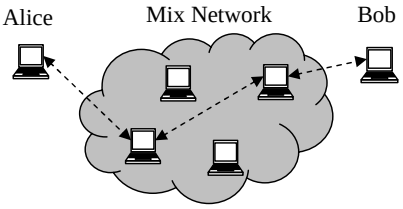

Fig. 1. Mix Network

In this paper, we will investigate the QoS of flow-based anonymity systems with several different configurations. In [8], a relatively complete list of batching strategies for a message-based mix is provided. Those strategies can be utilized to counter message-level (packet-level) timing attacks. In our opinion, not all of them are appropriate for flow-based systems. For example, in a *threshold mix*, a mix transmits a batch of packets only after the number of packets collected has exceeded a pre-defined threshold. This may cause serious problems for traffic of TCP flows. For example, if the first (SYN) packet of a TCP flow is collected by a mix which then waits indefinitely to reach its threshold, this may mean the SYN packet does not reach the receiver, the TCP flow never starts and the entire mix network is not stable. We have selected three batching and reordering strategies which seem to be feasible for a flow-based mix network and summarize them in Table I.

Many researchers have been studying possible attacks against flow-based anonymous communication networks [11], [12], [14], [21], [22]. There is little systematic analysis of the QoS of anonymous communication networks. In this paper, we study the QoS of TCP under the three batching and reordering strategies listed in Table I.

III. TCP Performance Degradation Analysis in flow-based Mix Networks

In this section, we analyze TCP performance degradation as the result of packet reordering caused by mix networks. We first discuss the false fast retransmit caused by one mix's batching and reordering schemes. Then we present the basic principle used to model TCP throughput. The success of modeling TCP throughput in a mix network lies in the estimation of the maximum congestion window. Finally, we extend the discussion to multiple mixes and present key observations.

A. False Fast Retransmit

A TCP connection transmits packets in bursts [23]. The number of packets sent in one burst is the instantaneous congestion window size $cwnd$ in the case of no packets dropped and a large enough receiver advertised window. An introduction to TCP congestion control can be found in Appendix A. In this paper, we adopt the common assumption from other TCP performance studies that both the receiver advertised

TABLE I
BATCHING AND REORDERING STRATEGIES [8]

Strategy Index	*Name*	*Adjustable Parameters*	*Algorithm*
S_0	Simple Proxy	$none$	No batching or reordering.
S_1	Timed Mix	$< t >$	A timer expires every t seconds. When it does, all the packets which have arrived since the last timer expiry get reordered with a random permutation and sent out.
S_2	Stop-and-go Mix (Continuous-time Mix)	$< \mu, \sigma^2 >$	Each packet is assigned a delay (deadline) drawn from an exponential distribution with mean μ and variance σ^2. A packet is sent out when its deadline is reached.

window and the receiver buffer are sufficiently large. Figure 2 illustrates burstiness of TCP Reno packets in our simulations. We can see that because of TCP's control mechanism, packets are sent out in bursts (the vertical segments of the graph)[1]. Note that the size of a burst corresponds to the current or *instantaneous* $cwnd$.

When a mix node receives a burst of packets from a sender, it may change the order of packets before forwarding them to the next mix or receiver. For example, the sender transmits packets 1, 2, 3, 4 and 5 in order while the receiver (after one or more mixes reorder the packets) may receive packets in the order 2, 3, 4, 5 and 1. A TCP receiver sends an immediate duplicate ACK whenever an out-of-order segment arrives. By the time packet 5 has been received, the receiver has already generated three duplicate ACKs, because packet 1 has not yet been received. *The three duplicate ACKs cause a false fast retransmit at the sender*, which assumes that the three duplicate ACKs signal a packet loss. The sender exercises the fast recovery and the TCP congestion avoidance process and cuts the TCP instantaneous $cwnd$ in half. Intuitively, such unnecessary retransmits will have a significant impact on TCP throughput in a mix network since the size of instantaneous $cwnd$ limits the maximum number of packets the TCP sender can send at one time. We will further analyze this impact by modeling the TCP throughput.

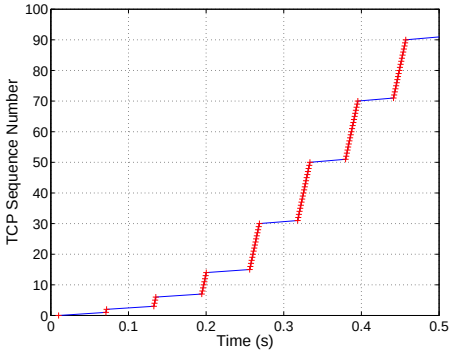

Fig. 2. Burstiness of TCP Reno Packets: a Mark "+" Refers to one TCP Packet from a Sender

B. Principle of Modeling TCP Throughput under One Mix

There exists a lot of outstanding work on TCP modeling such as [24]. Our modeling of TCP throughput in mix networks is based on the work by Sally Floyd and Kevin Fall [25]. Although the derived formulae are approximate, we found that they reflect the essence of batching and reordering's impact on TCP throughput. Our simulation results match the theoretical analysis well. In our study, we consider only the influence of false fast retransmit while limiting other factors such as packet drop. This is reasonable since this paper is addressing TCP performance as affected by batching and reordering strategies in a mix network. Batching and reordering is the dominant factor on TCP throughput when it is deployed in a mix network. Although packets drops are widespread on the Internet, the packet drop rate is relatively small [26].

Formula (1) from [25] gives an approximate estimate of TCP throughput in a normal network without batching and reordering:

$$T \leq \frac{0.75 \times W \times B}{R}. \quad (1)$$

where T is the throughput, W is the *maximum congestion window* (maximum $cwnd$) that a TCP connection can reach, B is the packet size, and R refers to the round trip time. To understand the maximum $cwnd$ W, recall that during the process of congestion avoidance, the increase in the instantaneous congestion window $cwnd$, denoted as w for simplicity, should be one segment each round-trip time. Therefore, w will increase linearly until duplicate ACKs or dropped packets cause TCP to enter congestion avoidance and cut w to half. Maximum $cwnd$ W is the maximum value of w before congestion avoidance divides it in half. In other words, W is a local maximum of w. Formula (1) assumes that in a stable state, TCP has a static maximum congestion window and round trip time. From (1), it is clear that the maximum congestion window W and the round trip time R control the TCP throughput.

In a mix network, things will be different: a mix will influence both the maximum congestion window (because of reordering) and the round trip time (because of batching). Both W and R in (1) will become random variables. From knowledge of statistics [27] (Formula (2.26)), if X and Y are independent random variables, then approximate expression for the mean of the quotient X/Y is given by

$$E(\frac{X}{Y}) = \frac{E(X)}{E(Y)}(1 + \frac{Var(Y)}{(E(Y))^2}). \quad (2)$$

Therefore, for mix networks, we may use formula (3) to estimate the average TCP throughput by calculating the means of W and R.

$$E(T) \leq \frac{0.75 \times E(W) \times B}{E(R)}(1 + \frac{Var(R)}{(E(R))^2}). \quad (3)$$

[1]This particular plot shows the slow start as well, since the bursts increase in size exponentially for the first 0.3 seconds, until the link bandwidth is fully utilized.

In (3), the mean of the round trip time $E(R)$ and the variance $Var(R)$ may be derived in theory and practice. For example, if one stop-and-go mix (refer to Table I) is used, $E(R) = 2\mu + E(\textit{queuing delay + transmission delay})$. The queuing delay and transmission delay can be estimated by techniques in [28], [29], [30], [31] and many others. Another assumption used to derive (3) is that the maximum congestion window W and round trip time R are independent. This assumption holds true since the maximum congestion window is majorly controlled by mix reordering and not related with the round trip time, as we can see later. Therefore, we will focus on deriving the mean of the maximum congestion window $E(W)$. W is a discrete random variable and it can have a value of 4 to ∞. W cannot be smaller than 4 because a window of less than 4 packets could not generate three duplicate ACKs and trigger false fast retransmit.

We now derive the formula to calculate the probability, $P(W = n)$, that W has a value of n. The fact that the maximum congestion window gets a value of n means that when the instantaneous $cwnd$ w is between $[1,\ n-1]$, the maximum number of duplicate ACKs, denoted as L, is less than or equal to 2; when the instantaneous congestion window w is n, $L \geq 3$ and the instantaneous $cwnd$ will be cut to half. Based on the total probability formula,

$$P(W = n) = P(\{w = 1, L \leq 2\}, \cdots, \{w = n, L \geq 3\}). \quad (4)$$

Since the influence of reordering at different w is independent, we have,

$$P(W = n) = P(w = n, L \geq 3) \prod_{k=1}^{n-1} P(w = k, L \leq 2). \quad (5)$$

As noted above, three duplicate ACKs are impossible when $w < 4$, so clearly $P(w = 1, L \leq 2) = 1$, $P(w = 2, L \leq 2) = 1$ and $P(w = 3, L \leq 2) = 1$. We will drop these three factors from the product in the following discussion.

Therefore, the mean of the maximum congestion window is

$$E(W) = \sum_{n=4}^{\infty} nP(W = n). \quad (6)$$

C. Mean of the Maximum Congestion Window

In this subsection, we will derive formulae for the mean of the maximum congestion window. We will first derive $P(w = i, L \leq 2)$ and $P(w = n, L \geq 3)$. Secondly, we will derive formulae for $P(W = n)$ in (5). We then derive the formulae for the mean of the maximum congestion window $E(W)$ by using the law of total probability (6). Finally, we present a few key observations based on the theorems.

1) Derivation of $P(w = i, L \leq 2)$ and $P(w = n, L \geq 3)$: To obtain the mean of the maximum congestion window $E(W)$, we need to derive $P(W = n)$ in (6), which requires the calculation of $P(w = i, L \leq 2)$ and $P(w = n, L \geq 3)$ (where $n \geq 4$) in (5). We need Lemma 1 to derive $P(w = i, L \leq 2)$ and $P(w = n, L \geq 3)$. The intuitive meaning of Lemma 1 is that if no packet is reordered and delayed two (or more) positions later, the reordering will not trigger the fast retransmit, and the resulting reduction in the instantaneous congestion window size.

Lemma 1: Given a window of n TCP packets $\{P_1,P_2,\ldots,P_n\}$ arriving at a mix. Let $f(i)$ be the index of P_i when the n packets pass through the mix after being randomly reordered. The sender will never receive more than two duplicate ACKs for any packet, if all packets satisfy

$$f(i) <= i + 2, i = 1, 2, \ldots, n. \quad (7)$$

Lemma 1 gives a very good estimation of how three duplicate ACKs are generated for a window of packets. Our simulation results in Section V verify the accuracy of Lemma 1. With the fact stated in Lemma 1, we can use the approach of permutations and combinations under the constraint from (7) to derive $P(w = i, L \leq 2)$ and $P(w = n, L \geq 3)$, as conducted in Lemma 2.

Lemma 2: Given a sequence of n packets $\{P_1,P_2,\ldots,P_n\}$, which are randomly reordered after passing a mix node, the probability that the new sequence does not cause *false fast retransmit* is given in (8) and the probability that the new sequence causes *fast retransmit* is given in (9).

$$P(w = n, L \leq 2) = \frac{3^{n-2} \times 2}{n!}. \quad (8)$$

$$P(w = n, L \geq 3) = 1 - \frac{3^{n-2} \times 2}{n!}. \quad (9)$$

Lemma 1 and Lemma 2 are basis for all the analysis in this paper. Their proof can be found in Appendix B.

2) Derivation of $P(W = n)$: Once $P(w = i, L \leq 2)$ and $P(w = n, L \geq 3)$ are derived, we can proceed to calculate $P(W = n)$ in (5) and we will have the following lemma.

Lemma 3: The probability that the *maximum congestion window* has a value of n can be derived from (10),

$$P(W = n) = \begin{cases} 0 & , 0 \leq n \leq 3; \\ 1 - \frac{3^{4-2} \times 2}{4!} & , n = 4; \\ \prod_{k=4}^{n-1} \frac{3^{k-2} \times 2}{k!} (1 - \frac{3^{n-2} \times 2}{n!}) & , n \geq 5\ . \end{cases} \quad (10)$$

and

$$\sum_{n=4}^{\infty} P(W = n) = 1. \quad (11)$$

3) Derivation of $E(W)$: Once Lemma 3 is derived, it is not difficult to derive the mean of the maximum congestion window in (6) and verify its convergence by appropriate arithmetic manipulation. Based on Lemma 3, we have Theorem 1.

Theorem 1: The mean of the maximum congestion window at duplicate threshold 3 is of convergence and can be derived as follows:

$$E(W) = \sum_{n=4}^{\infty} nP(W = n) = 8.27 \quad (12)$$

$$\leq \sum_{n=4}^{\infty} n \frac{3^{n-3} \times 2}{(n-1)!} = \frac{8}{9} e^3 - \frac{41}{9} = 13.30. \quad (13)$$

The proof is in Appendix C and the values in (12) and (13) were numerically derived using Matlab. Note that in deriving Lemma 2 and Lemma 3, we have taken an approximate

assumption that the instantaneous congestion window size w starts with 4 and increases until reaching the maximum congestion window W. According to the fast recovery algorithm in TCP Reno, the starting value of w at a sender is a random variable, which depends on the maximum congestion window size before 3 duplicate ACKs were received. When the mean of the maximum congestion window size is large, this approximation is pessimistic. In practice, our formulae give a close approximation of the actual maximum congestion window size. We will use simulations to demonstrate this claim in Section V. Our modeling also fits well with TCP congestion control in the 4.3BSD Tahoe release [32].

D. Observations and Extension to Multiple Mixes

We have two key observations on the preceding results: (1) In a mix network with batching and reordering applied, the TCP throughput is bounded regardless of the underlying physical network and link bandwidth. This poses a serious challenge for anonymous communication. Even with a deployed gigabit network, the TCP throughput in a mix network cannot be improved because the maximum congestion window is limited by the false fast retransmit caused by batching and reordering! (2) The above analysis applies to any batching and reordering strategy so long as a window of packets is reordered equiprobably, demonstrating the generality of our results.

We have discussed how to derive the mean of the maximum congestion window under a one-mix network. We can extend Formula (3) to a multiple-mix network. Approximately, we can view a multiple-mix network as an aggregate of one-mix networks and derive the TCP throughput as follows,

$$E(T) \leq \frac{0.75 \times E(W) \times B}{\sum_{k=1}^{m} E(R_k) + RTT}(1 + \frac{var(\sum_{k=1}^{m} R_k + RTT)}{(E(\sum_{k=1}^{m} R_k + RTT))^2}). \quad (14)$$

where m is the number of mixes a TCP connection passes through, $E(R_k)$ is the mean delay of a packet and the corresponding ACK buffered at the k^{th} mix, RTT is the two-way propagation delay, and $E(W)$ is the mean of the maximum congestion window under the aggregated one-mix network.

Assume that after passing through each mix, a window of packets are still clustered together. Then, it can be seen that the reordering of a window of packets is similar in a one-mix network and a multiple-mix network: each of the possible alignments of a window of packets is equiprobable. That is, the impact of the maximum congestion window on TCP performance is similar in a one-mix network and a multiple-mix network. Therefore, Formula (12) for the one-mix network case is still valid for the multiple-mix network case. However, because of the extra delay on each mix, the TCP performance would be further degraded.

IV. Improving TCP Performance in Mix Networks

From the discussion in Section III, we know that false fast retransmit is very likely to occur in a mix network with batching and reordering applied, which limits both the maximum congestion window and thus the TCP throughput. In this section, we examine methods to suppress false fast retransmits and improve TCP performance. As we know, fast retransmit requires a threshold for the number of duplicate ACKs that the sender must receive before it infers that the network has dropped a packet. This parameter, $dupthresh$, is fixed at 3 duplicate ACKs in the fast retransmit specification [16]. To suppress false fast retransmits, an intuitive approach is to increase the $dupthresh$ value. In the following, we analyze the impact of a larger $dupthresh$ on TCP throughput and discuss potential risks of this approach. We will also discuss other approaches for improving TCP performance in case of out-of-order packet delivery. In Section (V-C), we evaluate this method through simulations.

A. The Impact of $dupthresh$ on TCP Throughput

Denote D as a chosen duplicate threshold $dupthresh$. The mean of the maximum congestion window is given in (15),

$$E(W) = \sum_{n=D+1}^{\infty} nP(W = n). \quad (15)$$

where

$$P(W = n) = P(w = n, L \geq D) \prod_{k=D+1}^{n-1} P(w = k, L \leq D-1). \quad (16)$$

Theorem 2 gives the upper bound of $E(W)$, which is convergent given a duplicate threshold $dupthresh$ D. In fact, when D increases, $E(W)$ increases too. We have this verified in Theorem 3. Theorem 2 and Theorem 3 are the theoretical foundation of improving TCP performance by manipulating duplicate threshold in a mix network with batching and reordering. the detailed proof of these two theorems can be found in Appendix C.

Theorem 2: The mean of the maximum congestion window at duplicate threshold D is of convergence.

$$E(W) = \sum_{n=D+1}^{\infty} nP(W = n) \quad (17)$$

$$\leq \sum_{n=D+1}^{\infty} n\frac{D^{n-D} \times (D-1)!}{(n-1)!}. \quad (18)$$

Theorem 3: When the duplicate threshold D increases, the mean of the maximum congestion window $E(W)$ increases, so does the TCP throughput.

B. Cost of Increasing $dupthresh$

Theorem 3 demonstrates that increasing $dupthresh$ can improve TCP performance. However, there is cost of increasing $dupthresh$ [16]. While a larger $dupthresh$ value prevents the TCP sender from wrongly concluding that reorderings are losses, it also makes the TCP sender respond more slowly after real packet drops. When $dupthresh$ grows large, there are a number of risks that may cause TCP performance degradation.

When a real packet loss occurs, the TCP sender waits for duplicate ACKs to start fast retransmit and repair the

connection. If $dupthresh$ is too large, there may not be enough duplicate ACKs to activate fast retransmit and this will lead to generation of timeouts and incur longer packet delay. Even if enough duplicate ACKs return in the case of real packet loss, the fast retransmit will be delayed until all the required duplicate ACKs arrive at the sender. This will significantly increase the end-to-end delay for dropped packets. Some applications such as interactive transfers are intolerant of spikes in the end-to-end packet delay.

Delayed fast retransmit in the face of real packet loss will also delay the response of TCP to congestion. If a packet loss occurs during a short transfer where there are too few packets left to send to exceed $dupthresh$ and provoke a fast retransmit, this will dramatically increase the transfer time for small volume of data.

There is a clear tradeoff between avoiding false fast retransmits and the above-enumerated risks. A scheme for adapting $dupthresh$ must balance these opposing goals. Increasing $dupthresh$ alone is insufficiently adaptive; an algorithm for reducing $dupthresh$ is also needed. Zhang *et al.* [16] implemented an adaptive algorithm adjusting $dupthresh$ to deal with false fast retransmit while considering other factors such as real packet loss. We discuss the performance of their scheme in Section V-D.

Please note: one intuitive way to improve TCP performance in case of packet reordering is to buffer TCP packets and resequence them to make them in order before those packets get into the transport layer. Actually this is why TCP uses a buffer [33], whose size is *dupthresh*. The receiver's transport layer temporarily buffers out-of-order packets and re-sequence them as more packets arrive. So it is redundant and not necessary to add another buffer in front of the transport layer.

Another possible way to improve the performance of a mix network is to intelligently reorder packets. We could take a batch of messages (timed) and reorder it, while preserving the ordering of packets from the same source with respect to the other packets from the source (whether the source is a sender or mix). However, to use such an approach, a mix has to track all the flows passing through it, and the number of these flows and data requirements can be extremely high. This is analogous to the dilemma of integrated service vs. differential service. Although Tor [15] uses circuits to carry individual flows, this kind of "integrated service" has raised various privacy [22], [13] and performance issues [34]. Additionally, the scalability of Tor merits additional investigation as part of future research work.

V. Evaluation

In this section, we present simulation results to validate our analysis in section III and IV and to evaluate the $dupthresh$ based approach for improving TCP QoS in a mix network. These results are obtained by using the popular network simulation software ns-2 [35]. We use the *bootstrap matlab toolbox* [36] to generate the confidence interval for related figures in this section.

We have implemented different mix boxes with different batching and reordering strategies in ns-2. A mix box is an ns node that should be placed between sender and receiver nodes. With a mix box, packets entering the mix box can be batched and reordered based on the corresponding mix box type: MixBoxSG (continuous-time/stop-and-go mix) and MixBoxT (timed mix). A simple mix proxy behaves like a general router, except that all packets passing through it have the same size. The detailed introduction to ns-2 implementation of mixes is listed in Appendix E.

In the following, we will use the continuous-time mix as the example (as in [14]) to demonstrate how well simulation results match the theoretical analysis in Sections III and IV. Brief discussion on timed mixes and RR-TCP is given in Section V-D.

A. Simulation Setup

Figure 3 illustrates the simulation setup with the classical dumbbell topology, which is used for various TCP performance study. A sender and receiver communicate with each other through a series of mix boxes while there are crossover TCP sessions such as those from nodes S_n to R_n acting as background noise. In the case of a one-mix network, there is only one mix between the sender and receiver. We intentionally set the same bandwidth for all links, so there will be no bottleneck link and congestion. We set the queue size for all links to infinity. Therefore, if there is any fast retransmit, it is caused by batching and reordering strategies. We set the packet size to 512, corresponding to the setting in Tor, a popular anonymity application primarily targeting anonymous web browsing [15]. We set the advertised window for TCP flows to infinity. An FTP session is created between the sender and receiver. The above setting gives us an idealistic testbed to measure the influence of a mix box's batching and reordering strategy on TCP performance and this is the goal of this paper. In our simulation, the continuous-time mix has an average delay of 5 ms.

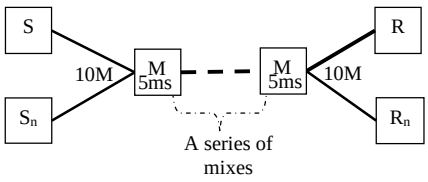

Fig. 3. Experiment Setup

B. TCP Performance Degradation in Mix Networks

To show the impact of the batching and reordering strategy on TCP performance, Figure 4 gives the changing trend of the congestion window ($cwnd$) with time in a normal network. That is, the mix box in Figure 3 is a normal node. In Figure 4, $cwnd$ keeps increasing because all the network links have the same bandwidth and the network has no congestion. Because of this monotonically increasing $cwnd$, the TCP throughput quickly increases in 0.4 seconds to the maximum speed, the link bandwidth of $10Mb/512/8 = 2441.4$ pkts/s as shown in Figure 5.

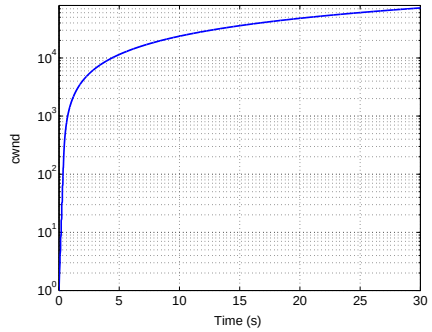

Fig. 4. Congestion Window in a Normal Network

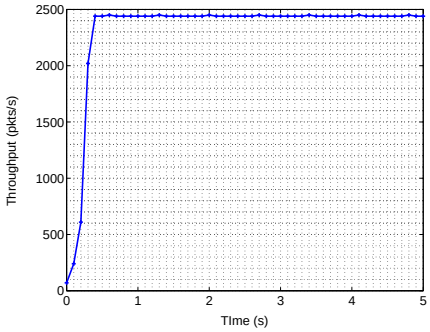

Fig. 5. TCP Throughput in a Normal Network

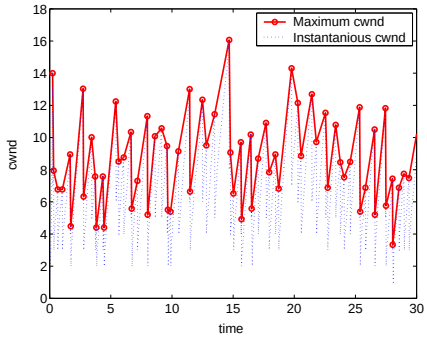

Fig. 6. Limited Maximum Congestion Window

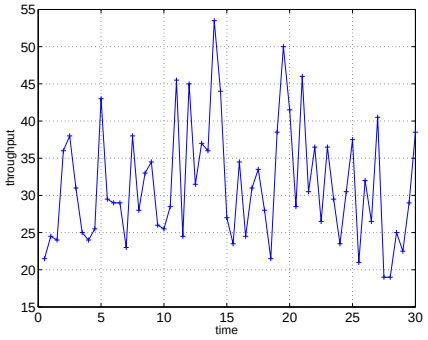

Fig. 7. Limited TCP Throughput Because of Limited Maximum Congestion Window

Figure 6 shows the changing trend of the congestion window ($cwnd$) with time. The dotted line is the instantaneous $cwnd$ changing with time and the bold solid line is the maximum $cwnd$ changing with time. Recall that the maximum $cwnd$ is the local maximum of the instantaneous $cwnd$. We can see that the maximum $cwnd$ is a random variable and it has a mean value, which can be calculated by (12). We can also derive the empirical from the measurement. The theoretical mean value of the maximum $cwnd$ is 8.27 and the empirical value is 8.76. We can see that the theorem matches reality very well. Figure 7 gives the throughput changing trend with time. If the mix box in Figure 3 is a normal node, the throughput will increase to the link bandwidth of $10Mb$ (2441.4 pkts/s) as shown in Figure 5. Because of reordering in the mix box, the mean throughput is dramatically reduced to 31.23 pkts/s as shown in Figure 7 and Figure 8.

Figure 8 illustrates the throughput in terms of the increasing link bandwidth. As we predicted in Section III-D, the TCP throughput does not change much with the increasing link bandwidth. This is because the mean of the maximum congestion window $E(W)$ has no relation with the underlying link bandwidth. In fact, the TCP throughput is slightly increasing with the increasing link bandwidth although the change is trivial. This is because the increasing bandwidth reduces the packet transmission delay and slightly reduces the round trip time $E(R)$.

Now we use simulation to verify the analysis in Section III-D for the case of multiple-mix networks. Figure 3 illustrates the multiple-mix network where multiple mixes are between the sender and receiver. Figure 9 shows $cwnd$ in terms of the number of mixes. As we analyzed in Section III-D, $cwnd$ does not go down with multiple mixes. Counter-intuitively, $cwnd$ actually increases a little bit with the increasing number of mixes. This is because with multiple mixes between a sender and a receiver, ACKs will be dispersed when they return to the sender. This reduces the chance that a window of packets are sent continuously at almost the same time from the sender, therefore reduces the chance that the mix box reorders those packets. But again, we can see that the increase in $cwnd$ is minor. Figure 10 shows that the round trip time RTT increases linearly with the increasing number of mixes, as we predicted. With an almost flat $cwnd$ and linearly increasing RTT, Figure 11 shows that the throughput reduces reciprocally, as predicted in (14). In the calculation of the theoretical throughput, $E(W) = 8.27$ based on (12) and observations and extension to multiple mixes in Section III-D, and the mean round trip time is the link propagation and average delay at mixes. We can see that the empirically measured throughput is bounded by the theoretically calculated one based on (14) and their trends match each other very well. The bound is loose because of the various approximations incorporated in the estimate, including the approximation used for deriving Formula (1) in [25].

C. Improving TCP Performance by Increasing $dupthresh$

In Section IV, we discussed improving TCP performance by increasing the duplicate ACK threshold $dupthresh$. The following simulation results are obtained in a one-mix network. Figure 12 shows the changing trend of the mean of the maximum congestion window in terms of the increasing $dupthresh$. Figure 13 shows the changing trend of the TCP

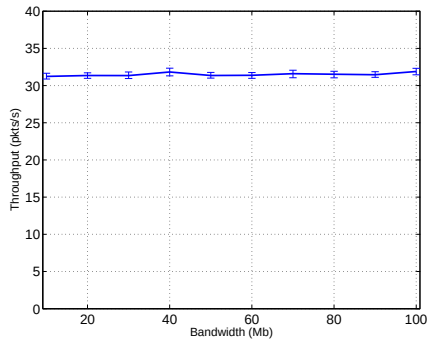

Fig. 8. Stable Throughput that Does Not Change with the Increasing Underlying Bandwidth

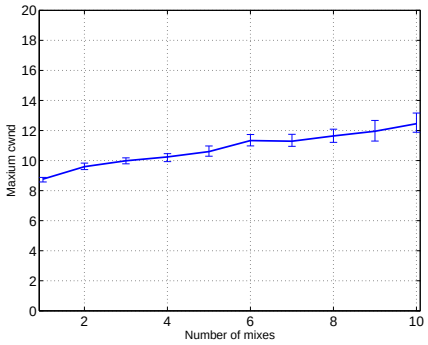

Fig. 9. cwnd Does Not Change Much with the Increasing Number of Mixes

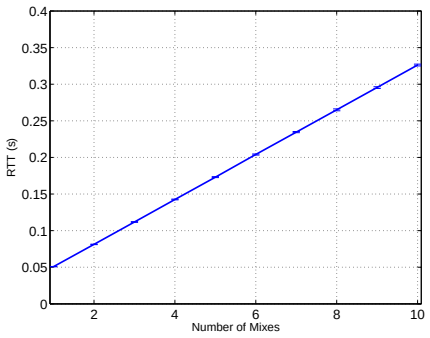

Fig. 10. RTT Increases Linearly with the Increasing Number of Mixes

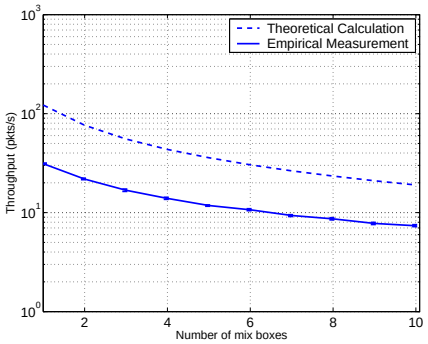

Fig. 11. TCP Throughput Decreases Quickly with the Increasing Number of Mixes

throughput in terms of the increasing $dupthresh$. From the two figures, we can make the following observations: By increasing $dupthresh$, we can increase TCP's maximum congestion window, as Theorem 3 predicts. Since the round trip time R does not change, increasing the maximum congestion window necessarily implies a corresponding increase in TCP throughput, as Formula (3) predicts and Figure 13 confirms.

As Figure 12 demonstrates, the theoretical curve of the mean of the maximum congestion window matches the empirical curve reasonably well. The difference between the two curves originates from our approximation approach, in which the starting congestion window that leads to the maximum congestion window is $dupthresh + 1$. This is not accurate as $dupthresh$ gets larger. The theoretical maximum $cwnd$ bound is loose because of approximations in our analysis.

In Figure 13, the TCP throughput reaches a maximum of 2440.9 pkts/s and stops growing as $dupthresh$ increases. This is because when the maximum $dupthresh$ reaches 192, the theoretical mean of the maximum $cwnd$ is large enough that it makes the TCP throughput reach the theoretic maximum possible speed, the link bandwidth of 2441.4 pkts/s. In this case, although $cwnd$ continues to grow because of the returning ACKs, the actual TCP throughput can not increase further.

D. Discussion

We have claimed that our analysis in Sections III and IV applies to any type of mix that randomly reorders packets. Figure 14 verifies this claim with a timed mix. In the figure, we show both analytical and empirical results of the mean of the maximum congestion window $cwnd$ which varies in terms of $dupthresh$ values. We can see that the measurements match the theoretical prediction given by Theorem 2 very well. Compared with continuous-time mix, the mean maximum congestion window size with a timed mix increases more sharply. This is because a timed mix flushes all packets in a window at the same time. Therefore, ACKs return to the sender sooner and this helps increase $cwnd$ quickly. In contrast, a continuous-time mix flushes a window of packets continuously. Recall that a continuous-time mix gives a random delay to each packet based on an exponential distribution.

In Section IV-B, we mentioned that the $dupthresh$ value should be adjusted adaptively at run time based on the current network state to achieve balance between the conflicting goals of suppressing false retransmits and mitigating the risks associated with larger $dupthresh$ values. RR-TCP [16] is such an algorithm, so we evaluated its performance in a mix network as the one in Figure 3 for the case of a one-mix network. In order to get the best performance, we optimized the parameters of the algorithm[2]. The results for RR-TCP in a continuous-time mix network are given in Figure 15, which presents the corresponding results of throughput. We make the following observations from Figure 15. RR-TCP can dynamically change the duplicate threshold. Because of the increasing $cwnd$, TCP

[2] $rtt\_his\_ = 1000.0$, $rtx\_bit\_ = 1$, $detectDSacks\_ = 1$, $dynamicThresh\_ = 1$, $avoidTimeout\_ = 1$, $newRtoEstimation\_ = 1$, $newStd\_ = 1$, $dynamicRttvarTimes\_ = 1$, $min\_thresh\_ = 3$, and $max\_thresh\_ = 100000$.

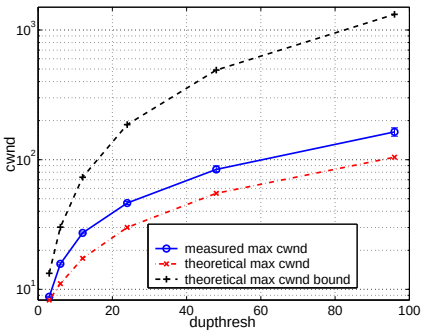

Fig. 12. cwnd Increases with the Increasing dupthresh

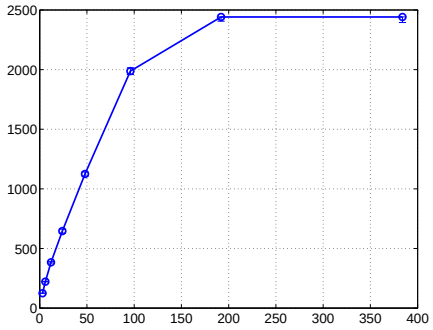

Fig. 13. Throughput Increases with the Increasing dupthresh

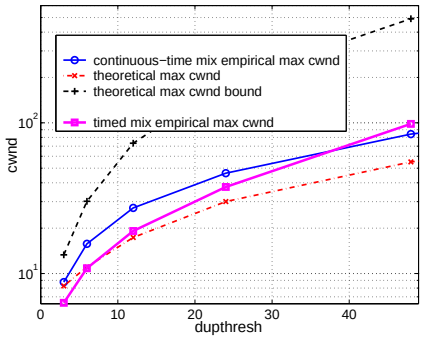

Fig. 14. cwnd Increases with the Increasing dupthresh for Timed Mix

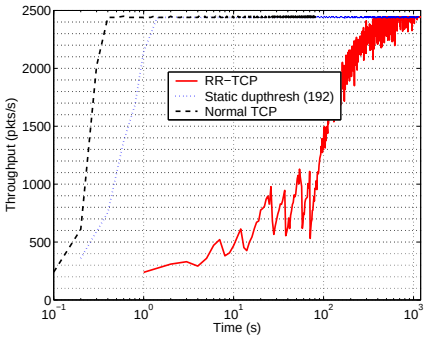

Fig. 15. Throughput of Continuous-time Mix in RR-TCP

throughput increases to the maximum. However, it takes a long time for RR-TCP to reach the maximum throughput, the link bandwidth. This transit time is around 500 seconds while it takes around 0.4 seconds for TCP Reno in a simple proxy mix network (*Normal TCP* in Figure 15) and around 1.6 seconds for TCP Reno with a static $dupthresh$ of 192 in a continuous-time mix network (*Static dupthresh (192)* in Figure 15) to reach the maximum throughput. This is because RR-TCP uses a loop control with feedback of SACKs and DSACKs, fast retransmit events and timeout events to adjust $dupthresh$. Apparently, the current implementation of this loop control lags well behind the network changes caused by the mix. Therefore, RR-TCP does not appear feasible for improving TCP performance in a mix network.

VI. Related Work

Attacks and Countermeasure in Anonymous Communication Systems: Since Chaum proposed the idea of anonymous computation and communication [4], researchers have applied the idea in various ways including message-based email systems and flow-based low-latency communications. Mix techniques can be used for either message-based (high-latency) or flow-based (low-latency) anonymity applications. Message-based email anonymity applications include the first Internet anonymity *remailer* by Helsingius [37], *cypherpunk remailer* by Eric Hughes and Hal Finney [38], *Babel* by Gülcü and Tsudik [39] and *Mixmaster* by Cottrell [40]. Danezis, Dingledine and Mathewson [5] recently developed a so-called Type III Anonymous Remailer Protocol *Mixminion*, whose design addresses a relatively complete set of attacks discovered by researchers thus far.

Low-latency anonymous communication can use either *core mix* networks or *peer-to-peer* networks. In a system using a core mix network, users connect to a pool of mixes and select a forwarding path through this core network to the receiver. *Tor* [15], *Onion routing* [41], *Freedom* [42] and many others belong to this category. In a peer-to-peer mix network, every node is a mix, but may also be a sender or receiver. A peer-to-peer mix network may scale well and provide better anonymity if enough participants use the anonymity service. *Crowds* [43], *Tarzan* [44], *ANODR* [45] and many others belong to this category.

QoS of Anonymous Communication: There is little systematic analysis of the quality of service (QoS) for anonymous communication networks. Rennhard *et al.* [46] empirically analyzed the performance of web browsing in a mix network, which uses a synchronized dummy message generation scheme. In their scheme, when a payload packet arrives at a mix and is ready for delivery to an output link, dummy messages are generated and delivered to other links to confuse the adversary. Fu *et al.* [11] provided an improved version of this scheme and performed empirical analysis.

Impact of Out-of-Order Packet Delivery on TCP: There are a number of works on how to improve TCP performance in the face of spurious retransmission. In an attempt to disambiguate duplicate ACKs caused by packet loss from those caused by packet reordering, the fast retransmission algorithm calls for the TCP sender to wait until three duplicate ACKs have arrived before retransmitting a segment [47]. Ludwig *et al.* [48] studied a scheme which uses TCP's time-stamp option and lets the sender time-stamp every packet sent. The

receiver echoes back the time-stamp in the corresponding ACK packets. Bohacek *et al*. [49] proposed a scheme that relies on timers to keep track of how long ago a packet was transmitted. Bhandarkar *et al*. [50] specified a set of TCP modification in the sender to disambiguate packet loss from reordering, using selective acknowledgements (SACKs) (given in RFC 2018) and the SACK-based loss recovery (given in RFC 3517). The basic idea of their scheme is to increase the threshold used to trigger a fast retransmission from the fixed value of three duplicate ACKs [47] to match the size of a congestion window. In order to improve the TCP performance in multi-hop mobile ad-hoc networks (MANETs) environment, Wang *et al*. [51] proposed a scheme to differentiate out-of-order packets from congestion loss, using additional sequence numbers (carried as TCP header options). Xia *et al*. [33] borrowed the idea of an automatic repeat request (ARQ) scheme in the network link layer and proposed a scheme in the receiver, using a specific buffer at the cost of extra delay for re-sequencing packets.

There are other works related to measuring packet reordering effects. For example, Piratla *et al*. [17] introduced a metric to capture the amount and degree of reordering. Bellardo *et al.* [18] studied a collection of active measurement techniques that can potentially estimate one-way end-to-end reordering rates to and from arbitrary TCP-based servers.

There is little theoretical analysis of the throughput of TCP due to the impact of false fast retransmit, which batching and reordering schemes will introduce. In this paper, we have systematically analyzed TCP performance in a flow-based mix network using batching and reordering schemes.

VII. Conclusion

This paper examined the degradation of TCP performance in flow-based mix networks incorporating batching and reordering techniques. Our theoretical analysis and simulation results demonstrate that TCP performance dramatically degrades in such a mix network. The reason is that TCP throughput has an approximately linear relationship with the mean of the maximum congestion window. Because of the out-of-order delivery caused by a mix's batching and reordering techniques, the mean of the maximum congestion window has a small, bounded value, which does not improve even with increases in the underlying link bandwidth. To improve TCP performance in such a flow-based mix network, we examined increasing TCP's duplicate threshold parameter, $dupthresh$. Our simulations show that we can improve TCP's maximum congestion window (and hence throughput) as our theorem predicts and this confirms the feasibility of the scheme. The numerical approximation of our theoretical curve for the mean of the maximum $cwnd$ matches the empirical curve well.

To the best of our knowledge, we are the first to develop a complete *theoretical model* for the impact of out-of-order delivery as in a mix network on TCP QoS and present formulae for approximate TCP throughput for such networks. We also analyzed the impact of increasing $dupthresh$ on TCP performance. Although there have been conjectures regarding the impact of batching and reordering on low latency anonymous communication, this paper gives the first formal proof of this impact and proposes an approach for improving performance when such anonymity schemes are applied.

Acknowledgments

This project was partially supported by NSF grants 0721766 and 0722856. Any opinions, findings, conclusions, and/or recommendations expressed in this material, either expressed or implied, are those of the authors and do not necessarily reflect the views of the sponsors listed above. The authors are grateful to the reviewers for their valuable comments that helped to improve the revised version. Part of this paper appeared in Proceedings of the IEEE International Symposium on Dependable, Autonomic and Secure Computing (DASC), Columbia, MD, September, 2007.

References

[1] Xinwen Fu, Bryan Graham, Dong Xuan, Riccardo Bettati, and Wei Zhao, "Empirical and theoretical evaluation of active probing attacks and their countermeasures," in *Proceedings of the 6th Information Hiding Workshop (IH)*, May 2004.

[2] D. X. Song, D. Wagner, and X. Tian, "Timing analysis of keystrokes and timing attacks on ssh," in *Proceedings of the 10th USENIX Security Symposium (SECURITY)*, August 2001.

[3] Qixiang Sun, Daniel R. Simon, Yimin Wang, Wilf Russell, Venkata N. Padmanabhan, and Lili Qiu, "Statistical identification of encrypted web browsing traffic," in *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, May 2002.

[4] David Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 4, no. 2, February 1981.

[5] George Danezis, Roger Dingledine, and Nick Mathewson, "Mixminion: Design of a type iii anonymous remailer protocol," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy (S&P)*, May 2003.

[6] Andrei Serjantov and George Danezis, "Towards an information theoretic metric for anonymity," in *Proceedings of Privacy Enhancing Technologies Workshop (PET)*, Roger Dingledine and Paul Syverson, Eds. April 2002, Springer-Verlag, LNCS 2482.

[7] Y. Guan, X. Fu, R. Bettati, and W. Zhao, "A quantitative analysis of anonymous communications," in *Proceedings of International Conference on Distributed Computing Systems (ICDCS)*, July 2002.

[8] A. Serjantov, R. Dingledine, and P. Syverson, "From a trickle to a flood: active attacks on several mix types," in *Proceedings of Information Hiding Workshop (IH)*, February 2002.

[9] X. Wang, S. Chen, , and S. Jajodia, "Tracking anonymous peer-to-peer voip calls on the internet," in *Proceedings of the 12th ACM Conference on Computer Communications Security (CCS)*, November 2005.

[10] P. Peng, P. Ning, and D. S. Reeves, "On the secrecy of timing-based active watermarking trace-back techniques," in *Proceedings of the IEEE Security and Privacy Symposium (S&P)*, May 2006.

[11] Ye Zhu, Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao, "On flow correlation attacks and countermeasures in mix networks," in *Proceedings of Workshop on Privacy Enhancing Technologies (PET)*, May 2004.

[12] Xinwen Fu, Y. Zhu, B. Graham, R. Bettati, and Wei Zhao, "On flow marking attacks in wireless anonymous communication networks," in *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, April 2005.

[13] Wei Yu, Xinwen Fu, Steve Graham, Dong Xuan, and Wei Zhao, "DSSS-based flow marking technique for invisible traceback," in *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, May 2007.

[14] G. Danezis, "The traffic analysis of continuous-time mixes," in *Proceedings of Workshop on Privacy Enhancing Technologies (PET)*, May 2004.

[15] Roger Dingledine, Nick Mathewson, and Paul Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium (SECURITY)*, August 2004.

[16] Ming Zhang, Brad Karp, Sally Floyd, and Larry Peterson, "RR-TCP: A reordering-robust TCP with DSACK," in *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP)*, November 2003.

[17] N. M. Piratla, A. P. Jayasumana, and A. A. Bare, "Reorder density (rd): a formal, comprehensive metric for packet reordering," in *Proceedings of the 30th Local Computer Network Conference (LCN)*, November 2005.

[18] J. Bellardo and S. Savage, "Measuring packet reordering," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment (IMW)*, November 2002.

[19] Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields, "An analysis of the degradation of anonymous protocols," in *Proceedings of the Network and Distributed Security Symposium (NDSS)*, February 2002.

[20] M. Wright, M. Adler, B. N. Levine, and C. Shields, "Defending anonymous communication against passive logging attackss," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy (S&P)*, May 2003.

[21] Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew Wright, "Timing attacks in low-latency mix-based systems," in *Proceedings of Financial Cryptography (FC)*, February 2004.

[22] Steven J. Murdoch and George Danezis, "Low-cost traffic analysis of tor," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy (S&P)*, May 2005.

[23] W. Richard Stevens, *TCP/IP Illustrated, Volume 1 The Protocols*, Addison-Wesley Professional, 1999.

[24] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe, "Modeling TCP throughput: A simple model and its empirical validation," in *Proceedings of ACM SIGCOMM*, September 1998.

[25] Sally Floyd and Kevin Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 458–472, 1999.

[26] Vern Paxson, "End-to-end Internet packet dynamics," *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 277–292, 1999.

[27] Dennis Blumenfeld, *Operations Research Calculations Handbook*, CRC, 2001.

[28] V. JACOBSON, "Congestion avoidance and control," in *Proceedings of SIGCOMM*, 1988.

[29] P. Karn and C. Partridge, "Improving round-trip time estimates in reliablie transport protocols," in *Proceedings of SIGCOMM*, September 1987.

[30] C. Parsa and J.J. Garcia-Luna-Aceves, "Improving TCP congestion control over internets with heterogeneous transmission media," in *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, October 1999.

[31] Sharad Jaiswal, Gianluca Iannaccone, Christophe Diot, Jim Kurose, and Don Towsley, "Inferring tcp connection characteristics through passive measurements," in *Proceedings of Infocom*, June 1994.

[32] Kevin Fall and Sally Floyd, "Simulation-based comparisons of Tahoe, Reno and SACK TCP," *Computer Communication Review (ACM)*, vol. 26, no. 3, pp. 5–21, July 1996.

[33] Y. Xia and D. Tse, "An analysis on packet resequencing for reliable network protocols," in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom)*, March-April 2003.

[34] Damon McCoy, Kevin Bauer, Dirk Grunwald, Parisa Tabriz, and Douglas Sicker, "Shining light in dark places: A study of anonymous network usage," Tech. Rep., University of Colorado at Boulder, August 2007.

[35] VINT project, "The network simulator - ns-2," http://www.isi.edu/nsnam/ns/.

[36] Abdelhak M. Zoubir and D. Robert Iskander, "Bootstrap toolbox," http://www.csp.curtin.edu.au/downloads/bootstrap_toolbox.html, 2006.

[37] Johan Helsingius, "Press release: Johan helsingius closes his internet remailer," http://www.penet.fi/press-english.html, 1996.

[38] Sameer Parekh, "Prospects for remailers - where is anonymity heading on the internet," http://www.firstmonday.dk/issues/issue2/remailers/, 1996.

[39] Ceki Gülcü and Gene Tsudik, "Mixing E-mail with Babel," in *Proceedings of the Network and Distributed Security Symposium (NDSS)*, February 1996.

[40] Ulf Möller and Lance Cottrell, "Mixmaster Protocol — Version 2," http://www.eskimo.com/~rowdenw/crypt/Mix/draft-moeller-mixmaster2-protocol-00.txt, January 2000.

[41] P. F. Syverson, D. M. Goldschlag, and M. G. Reed, "Anonymous connections and onion routing," in *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, May 1997.

[42] Philippe Boucher, Adam Shostack, and Ian Goldberg, "Freedom systems 2.0 architecture," in *Zero Knowledge Systems, Inc.*, December 2000.

[43] Michael Reiter and Aviel Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security*, vol. 1, no. 1, 1998.

[44] Michael J. Freedman and Robert Morris, "Tarzan: A peer-to-peer anonymizing network layer," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*, November 2002.

[45] Jiejun Kong and Xiaoyan Hong, "ANODR: anonymous on demand routing with untraceable routes for mobile ad-hoc networks," in *Proceedings of the 4th ACM international symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc)*, June 2003.

[46] M. Rennhard, S. Rafaeli, L. Mathy, B. Plattnet, and D. Hutchison, "Analysis of an anonymity network for web browsing," in *Proceedings of the IEEE the 11th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE)*, September 2002.

[47] M. Allman, V. Paxson, and W. Stevens, "Tcp congestion control," *RFC 2581*, 1999.

[48] R. Ludwig and R. Katz, "The eifel algorithm: making tcp robust against spurious retransmission," *ACM Computer Communication Review*, vol. 30, no. 1, pp. 30–36, 2000.

[49] S. Bohacek, J. P. Hespanha, J. Lee, C. Lim, and K. Obraczka, "Tcp-pr tcp for persistent packet reordering," in *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, May 2003.

[50] S. Bhandarkar, A. L. N. Reddy, M. Allan, and E. Blanton, "Improving the robustness of tcp to non-congestion events," *RFC 4653*, 2006.

[51] F. Wang and Y. Zhang, "Improving tcp performance over mobile ad-hoc networks with out-of-order detection and response," in *Proceedings of the 3th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, June 2002.

APPENDIX A - TCP CONGESTION CONTROL

Modern implementations of TCP must contain four intertwined algorithms related to congestion control: slow start, congestion avoidance, fast retransmit, and fast recovery. Figure 16 illustrates the state transition between them.

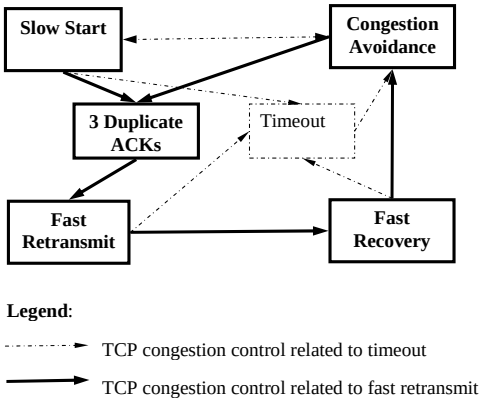

Fig. 16. TCP Reno Congestion Control

A TCP connection uses two variables to control the number of packets injected into the network: the receiver's advertised window and the congestion avoidance window ($cwnd$). The TCP output routine never sends more than the minimum of $cwnd$ and the receiver's advertised window. $cwnd$ is based on the sender's assessment of network congestion; the receiver's advertised window is related to the amount of available buffer space at the receiver for this connection. Below we give a brief description on how the four algorithms in Figure 16 use these two variables.

To avoid congesting intermediate routers, which normally have limited buffer space, a newly established TCP connection uses the slow start algorithm. $cwnd$ is initialized to one segment. Each time an ACK is received, $cwnd$ is increased

by one segment. So, after sending one segment successfully, the sender transmits two segments, and if they are both acknowledged, the sender will send four segments, and so on. Therefore, $cwnd$ increases exponentially with time, and so does the number of packets transmitted over the connection. At some point an intermediate router will start discarding packets because of buffer limitations. The sender receives no ACKs and infers that its congestion window has become too large. There are two indications of packet loss: a timeout occurring at the sender and the receipt of duplicate ACKs at the sender.

Whenever there is a timeout, TCP always enters congestion avoidance, which leads to slow start again. When a TCP connection starts congestion avoidance because of packet loss, one-half of the current window size[3] is saved in the slow start threshold size, $ssthresh$, and $cwnd$ is set to one segment. Therefore the TCP connection uses the slow start algorithm to restart and $cwnd$ increases exponentially. However, once $cwnd$ is greater than $ssthresh$, the increase in $cwnd$ will be limited to at most one segment each round-trip time (RTT) regardless of how additional segments are acknowledged during that RTT.

A TCP receiver sends an immediate duplicate ACK when an out-of-order segment arrives. The purpose of this ACK is to inform the sender that a segment was received out-of-order and which sequence number is expected. *Three* duplicate ACKs[4] may indicate a packet loss. Whenever the packet loss is indicated by three duplicate ACKs, the sender uses fast retransmit to send the missing packet, without waiting for a retransmission timer to expire. Then the fast recovery starts to keep the network pipe full. The TCP connection sets $ssthresh$ to one-half $cwnd$, but no less than two segments. Please refer to [47] for details of the fast recovery algorithm. After fast retransmit sends what appears to be the missing segment and fast recovery finishes, congestion avoidance, not slow start, is performed. The reason for not performing slow start in this case is that the receipt of the duplicate ACKs tells TCP more than just a packet loss: since the receiver can only generate a duplicate ACK when another segment is received, presumably the receiver has successfully received and buffered that segment. When the fast recovery is finished, the TCP connection sets $cwnd$ to $ssthresh$ and this leads to congestion avoidance. The $cwnd$ will be reset to $ssthresh$ and increase by one each RTT, but the $cwnd$ will not be reset to 1 as in a slow start.

APPENDIX B

Lemma 1. *Given a window of n TCP packets $\{P_1,P_2,\ldots,P_n\}$ arriving at a mix. Let $f(i)$ be the index of P_i when the n packets pass through the mix after being randomly reordered. Under these conditions, the sender will never receive more than two duplicate ACKs for any packet, if all packets satisfy*

$$f(i) <= i+2, i = 1, 2, \ldots, n. \tag{19}$$

Proof:

As Figure 2 shows, because of TCP's congestion control mechanism, packets are always sent out in bursts of $cwnd$ size. To simplify the discussion, we assume that one window of packets is sent out at almost the same time, so it will be collected as and treated as part of a single batch. When the number of packets within one window is reasonably small, this assumption holds well. This is the case within our context since a mix's reordering mechanism prevents the size of the instantaneous $cwnd$ from getting very large.

We consider two cases for a packet P_i satisfying (19), in order to prove that if formula (19) is satisfied, the reordering of P_i to position $f(i)$ will not generate three duplicate ACKs for P_i.

Case 1: $f(i) <= i$. In this case, packet P_i arrives earlier than or exactly when it is expected by the receiver. The receiver will not send a duplicate ACK asking for P_i since the receiver must still be waiting for some packet P_j, $j < i$ to arrive.

Case 2: $f(i) = i+1$ *or* $i+2$. In this case, the i^{th} packet that the receiver receives is not P_i and P_i is not among the packets it has already received. If the receiver is waiting on any earlier packet than P_i, no duplicate ACK will be issued for P_i, if all earlier packets have already been received, the receiver sends a duplicate ACK for P_i. Then, if $f(i) = i+1$, then the next packet that the receiver receives is just P_i and it does not need to send a second duplicate ACK for P_i. If $f(i) = i+2$, then the receiver will send a second duplicate ACK but no more. Either way, the sender will receive at most two duplicate ACKs for P_i, and thus not trigger "fast retransmit".

Figure 17 gives an extreme case still satisfying the constraint (19) where the window size is 6, the last two packets are reordered to the first positions, and earlier packets are reordered to later positions, $f(6) = 1$, $f(5) = 2$, $f(1) = 3$, $f(2) = 4$, $f(3) = 5$, and $f(4) = 6$. *Note: P_1 cannot be reordered to a position later than 3 because of the constraint from (19).* For all packets except the first two, $f(i) = i+2$. Based on the TCP ACK generation mechanism, a TCP receiver sends an immediate duplicate ACK when an out-of-order segment arrives and notifies the sender what sequence number is expected. When P_6 reaches the receiver, which is waiting for P_1, the receiver generates one duplicate ACK for P_1. When P_5 arrives, the receiver generates the second ACK asking the sender for P_1. Then P_1 arrives, the receivers generates one ACK asking for P_2. As we go through this process of ACK generation, we can see that there will be no more than two duplicate ACKs generated for any packet in this extreme case.

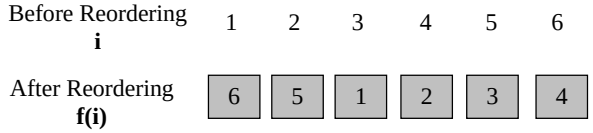

Fig. 17. An Extreme Case of Reordering

■

Lemma 2. *Given a sequence of n packets $\{P_1,P_2,\ldots,P_n\}$, which are randomly reordered after passing a mix node, the probability that the new sequence does not cause false fast*

[3]The minimum of $cwnd$ and the receiver's advertised window, but at least two segments.

[4]Four identical ACKs without the arrival of any other intervening packet.

retransmit is given in (8) and the probability that the new sequence causes fast retransmit is given in (9).

$$P(w=n, L\le 2) = \frac{3^{n-2}\times 2}{n!}. \quad (20)$$

$$P(w=n, L\ge 3) = 1-\frac{3^{n-2}\times 2}{n!}. \quad (21)$$

Proof:

Now we derive the probability, $P(w=n, L\le 2)$, that all packets satisfy (19) in a window of n TCP packets. We know that the number of permutations of n packets is $n!$. We need to know how many among them satisfy (7), i.e., the number of "valid" sequences. Let us try to build a valid sequence and count the number of choices for each packet's position in the sequence. We denote the *old sequence* as the sequence of packets going into a mix and the *new sequence* as the sequence of the same set of packets leaving the mix. Consider P_1, the first packet in the old sequence. In the new sequence, it is allowed to be at position $f(1)=1,\ 2,\ or\ 3$, given the constraint (7). So there are 3 possibilities. Then consider P_2, which can be placed at position $1, 2, 3\ or\ 4$ in the new sequence. Since P_1 already occupies one position, P_2 may choose any of the remaining 3 positions. It is not difficult to verify that, for any packet $P_i, i\le n-1$, once the positions of P_1, P_2, ..., P_{i-1} are decided, there are only 3 choices for P_i's position, i.e. $f(i)$. For packets P_{n-1} and P_n, there are 2 and 1 possibilities for $f(n-1)$ and $f(n)$, respectively, because $n+1$ and $n+2$ are not valid positions in a new sequence.

As Figure 17 shows: once P_1, P_2, P_3 and P_4 take the positions of 3, 4, 5 and 6 in the new sequence, P_5 can only take positions of 1 or 2. Once P_5 takes position 2, P_6 can only take position 1. Therefore, the number of different valid sequences of n packets is $3^{n-2}\times 2\times 1$. The probability of a random sequence of n packets being valid is

$$P(w=n, L\le 2) = \frac{3^{n-2}\times 2}{n!}. \quad (22)$$

Since $P(w=n, L\ge 3) = 1-P(w=n, L\le 2)$, (9) can be derived from (8) directly. ■

APPENDIX C

Theorem 1. *The mean of the maximum congestion window at duplicate threshold 3 is of convergence.*

$$E(W) = \sum_{n=4}^{\infty} nP(W=n) = 8.27 \quad (23)$$

$$\le \sum_{n=4}^{\infty} n\frac{3^{n-3}\times 2}{(n-1)!} = \frac{8}{9}e^3 - \frac{41}{9} = 13.30. \quad (24)$$

Proof: From (10), we know

$$P(W=n) = \prod_{k=4}^{n-1}\frac{3^{k-2}\times 2}{k!}(1-\frac{3^{n-2}\times 2}{n!}) \quad (25)$$

$$= \prod_{k=4}^{n-2}\frac{3^{k-2}\times 2}{k!}(1-\frac{3^{n-2}\times 2}{n!})\frac{3^{n-3}\times 2}{(n-1)!}. \quad (26)$$

We can see that when $k\ge 4$, $\frac{3^{k-2}\times 2}{k!} < 1$ and $1-\frac{3^{n-2}\times 2}{n!} < 1$, therefore

$$P(W=n) \le \frac{3^{n-3}\times 2}{(n-1)!}. \quad (27)$$

Then we have

$$E(W) = \sum_{n=4}^{\infty} nP(W=n) \quad (28)$$

$$\le \sum_{n=4}^{\infty} n\frac{3^{n-3}\times 2}{(n-1)!}. \quad (29)$$

We know

$$e^x = \sum_{k=0}^{\infty}\frac{x^k}{k!}. \quad (30)$$

Therefore,

$$e^3 = \sum_{k=0}^{\infty}\frac{3^k}{k!}. \quad (31)$$

With appropriate manipulation of (29) with the knowledge of (31), we can have

$$E(W) \le \sum_{n=4}^{\infty} n\frac{3^{n-3}\times 2}{(n-1)!} \quad (32)$$

$$= \sum_{n=4}^{\infty}(n-1+1)\frac{3^{n-3}\times 2}{(n-1)!} \quad (33)$$

$$= \sum_{n=4}^{\infty}\frac{3^{n-3}\times 2}{(n-2)!} + \sum_{n=4}^{\infty}\frac{3^{n-3}\times 2}{(n-1)!} \quad (34)$$

$$= \frac{8}{9}e^3 - \frac{41}{9} \quad (35)$$

$$= 13.30. \quad (36)$$

Using the mathematical tool *maple*, we can also determine a numeric approximation to $E(W)$,

$$E(W) \approx 8.27. \quad (37)$$

■

APPENDIX D

Theorem 2. *The mean of the maximum congestion window at duplicate threshold D is of convergence.*

$$E(W) = \sum_{n=D+1}^{\infty} nP(W=n) \quad (38)$$

$$\le \sum_{n=D+1}^{\infty} n\frac{D^{n-D}\times (D-1)!}{(n-1)!}. \quad (39)$$

Proof:

Following the same deductive approach used in Lemma 2, we derive formulae (40) and (41) corresponding to (8) and (9),

$$P(w=n, L\le D-1) = \frac{D^{n-(D-1)}\times (D-1)!}{n!}. \quad (40)$$

$$P(w=n, L\ge D) = 1-\frac{D^{n-(D-1)}\times (D-1)!}{n!}. \quad (41)$$

Therefore, substitute (40) and (41) into (16), we have (42) corresponding to (10),

$$P(W=n)=\begin{cases} 0 & ,0\le n\le D; \\ 1-\frac{D^{D+1-(D-1)}\times(D-1)!}{(D+1)!} & ,n=D+1; \\ \prod_{k=D+1}^{n-1}\frac{D^{k-(D-1)}\times(D-1)!}{k!} & \\ \quad(1-\frac{D^{n-(D-1)}\times(D-1)!}{n!}) & ,n\ge D+2\ . \end{cases} \tag{42}$$

Substitute (42) into (15). By using a similar deductive approach as in Theorem 1, we have (18).

Now we prove that the right term of (18) is convergent. Using the ratio test to verify the convergence of $E(W)$, we find:

$$\rho=\lim_{n\to\infty}=\frac{\mu_{n+1}}{\mu_n}, \tag{43}$$

where

$$\mu_n=n\frac{D^{n-D}\times(D-1)!}{(n-1)!}. \tag{44}$$

Therefore

$$\begin{aligned} \rho &= \lim_{n\to\infty}\frac{(n+1)\frac{D^{n+1-D}\times(D-1)!}{(n+1-1)!}}{n\frac{D^{n-D}\times(D-1)!}{(n-1)!}} && (45)\\ &= \lim_{n\to\infty}D\frac{n+1}{n}\frac{1}{n} && (46)\\ &= 0. && (47)\end{aligned}$$

Since $\rho<1$, $E(W)$ converges. ■

Theorem 3. *When the duplicate threshold D increases, the mean of the maximum congestion window $E(W)$ increases, so does the TCP throughput.*

Proof: To prove this theorem, we rewrite (42) in the case of $n\ge D+2$.

$$\begin{aligned} P(W=n) &= \prod_{k=D+1}^{n-1}\frac{D^{k-(D-1)}\times(D-1)!}{k!} \\ &\quad\times(1-\frac{D^{n-(D-1)}\times(D-1)!}{n!}) && (48)\\ &= \prod_{k=D+1}^{n-1}\frac{D^{k-(D-1)}\times(D-1)!}{k!} \\ &\quad-\prod_{k=D+1}^{n}\frac{D^{k-(D-1)}\times(D-1)!}{k!}. && (49)\end{aligned}$$

Let's define

$$g(n,D)=\prod_{k=D+1}^{n}\frac{D^{k-(D-1)}\times(D-1)!}{k!}. \tag{50}$$

Then $\forall n\ge D+2$,

$$P(W=n)=g(n-1,D)-g(n,D). \tag{51}$$

It is easy to see that when $n=D+1$, we have

$$P(W=n)=1-g(n,D). \tag{52}$$

Denote $E(W,D)$ as the mean of the maximum congestion window when the duplicate threshold is D,

$$\begin{aligned} E(W,D) &= \sum_{n=D+1}^{\infty}nP(W=n) && (53)\\ &= (D+1)(1-g(D+1,D))+(D+2) \\ &\quad(g(D+2-1,D)-g(D+2,D)) \\ &\quad+\cdots && (54)\\ &= D+1+g(D+1,D)+ \\ &\quad g(D+2,D)+\cdots && (55)\\ &= D+1+\sum_{n=D+1}^{\infty}g(n,D). && (56)\end{aligned}$$

Now let's prove $E(W,D+1)>E(W,D)$

$$E(W,D+1)=D+2+\sum_{n=D+2}^{\infty}g(n,D+1). \tag{57}$$

Therefore, if we can prove each item of $E(W,D+1)$ is greater than each item of $E(W,D)$, then $E(W,D+1)>E(W,D)$. Clearly the first item, $D+2$, of $E(W,D+1)$ is greater than the first item, $D+1$, of $E(W,D)$. Now let's prove for other items, $g(n,D+1)>g(n,D)$.

$$\begin{aligned} g(n,D+1) &= \prod_{k=D+2}^{n}\frac{D^{k-(D+1-1)}\times(D+1-1)!}{k!} && (58)\\ &= \prod_{k=D+2}^{n}\frac{D^{k-D}\times D!}{k!}. && (59)\\ g(n,D) &= \prod_{k=D+1}^{n}\frac{D^{k-(D-1)}\times(D-1)!}{k!} && (60)\\ &= \frac{D}{D+1}\prod_{k=D+2}^{n}\frac{D^{k-(D-1)}\times(D-1)!}{k!} && (61)\\ &= \frac{D}{D+1}\prod_{k=D+2}^{n}\frac{D^{k-D}\times D!}{k!}. && (62)\end{aligned}$$

To prove $g(n,D+1)>g(n,D)$, we only need to prove $g(n,D+1)/g(n,D)\ge 1$

$$\begin{aligned} \frac{g(n,D+1)}{g(n,D)} &= \frac{D+1}{D} && (63)\\ &> 1. && (64)\end{aligned}$$

Therefore, $E(W,D+1)>E(W,D)$ and the mean of the maximum congestion window $E(W)$ increases when the duplicate threshold D increases. Moreover, since the mean of the round trip time in (3) does not change, an increasing $E(W)$ implies an increasing TCP throughput.

■

APPENDIX E - MIX IMPLEMENTATION IN NS-2

Node/MixBoxSG consists of a Classifier/MixBoxSG that sits in front of the default classifier, delays and reorders packets, and sends packets in batches at fixed intervals. When a packet is sent, Classifier/MixBoxSG passes the packet on to the default classifier. When a packet arrives at the mix box, a

deadline is generated from an exponential distribution for this packet, which is put into a buffer. Packets are sorted on the packet deadline and the packet with the earliest deadline is at the beginning of the buffer. A timer event is generated for this first packet in the buffer. If a new packet's deadline is earlier than the deadline of the first packet in the buffer, we cancel the current timer event and schedule a new one for the new packet.

Node/MixBoxT also consists of a Classifier/MixBoxT that sits in front of the default classifier, delays and reorders packets, and sends packets in batches at fixed intervals. Packets are stored in a buffer. The packet buffer is implemented as a *multi-map*, in which the key is a generated value from a uniform distribution. A timer expires periodically that flushes packets in the buffer out. When a packet is sent, Classifier/MixBoxT passes the packet on to the default classifier.

Dr. Xinwen Fu is an assistant professor in the Department of Computer Science, University of Massachusetts Lowell. He received his B.S. (1995) and M.S. (1998) in Electrical Engineering from Xi'an Jiaotong University, China and University of Science and Technology of China respectively. He obtained his Ph.D. (2005) in Computer Engineering from Texas A&M University. From 2005 to 2008, he was an assistant professor with the College of Business and Information Systems at Dakota State University. In summer 2008, he joined University of Massachusetts Lowell as a faculty member.

Dr. Fu won the 2nd place in the graduate category of the International ACM student research contest in 2002, the Graduate Student Research Excellence Award of the Department of Computer Science at Texas A&M University in 2004, the Merrill Hunter Award for Excellence in Research at Dakota State University in 2008 and the best paper award at International Conference on Communications (ICC) 2008.

Dr. Fu's current research interests are in network security and privacy, information assurance, computer forensics, system reliability and networking QoS. Dr. Fu has been publishing papers in conferences such as IEEE Symposium on Security and Privacy (S&P), IEEE International Conference on Computer Communications (INFOCOM) and IEEE International Conference on Distributed Computing Systems (ICDCS), journals such as IEEE Transactions on Parallel and Distributed Systems (TPDS), and book chapters. His research is supported by NSF.

Dr. Wei Yu received his Ph.D. degree in Computer Engineering from Texas A&M University in May 2008. He received his B.S. degree in Electrical Engineering from Nanjing University of Technology and his M.S. degree in Electrical Engineering from Tongji University. Since May 2001, he has been also working for Cisco Systems, Inc. His research interests are in the areas of computer/network security, information assurance, networking technologies, and distributed systems. Dr. Yu received the best paper award at International Conference on Communications (ICC) 2008. He has been publishing papers in conferences such as IEEE Symposium on Security and Privacy (S&P) and IEEE International Conference on Computer Communications (INFOCOM), journals such as IEEE Transactions on Parallel and Distributed Systems (TPDS), and book chapters.

Dr. Shu Jiang received his B.E. degree from University of Science and Technology of China in 1990, M.E. degree from Nanjing University in 1993, and Ph.D degree in Computer Science from Texas A&M University in 2005. He is currently working in Bridge360, Inc. His research and development interests include network security, wireless and mobile networking, parallel and distributed system, object-oriented software engineering, etc.

Dr. Steve Graham have been working with computers for over 30 years and has a strong mix of industrial and academic experience. He has worked for Hewlett-Packard (big company), Cadence Design Systems (medium), and a number of startups (small). He has taught at the University of Missouri-Kansas City, the University of Kansas, and now Dakota State University. His main interest has always been AI and learning related, but he has also working on written compilers and other embedded systems tools; Dr. Graham worked on teams developing discrete event simulation tools and on teams building a database engine and analytic tools from scratch. He has programmed in about 40 languages, give or take a half dozen depending on how you count them.

Dr. Yong Guan is the Litton assistant professor in the Department of Electrical and Computer Engineering at Iowa State University. He received his B.S and M.S degrees in Computer Science from Peking University, China, in 1990 and 1996, respectively, and his Ph.D. degree in Computer Science from Texas A&M University in 2002. Between 1990 and 1997, he worked as an assistant engineer (1990-1993) and lecturer (1996-1997) in Networking Research Group of Computer Center at Peking University, China. In 2002, he joined Iowa State University as a faculty.

His current research interests are in security issues in computer networks and distributed systems, including computer and network forensics, wireless and sensor network security, and privacy-enhancing technologies for the Internet. Yong Guan received the best student paper award from the IEEE National Aerospace and Electronics Conference in 1998, the 2nd Place Winner in graduate category of the International ACM student research contest in 2002, NSF Career Award in 2007, Iowa State University Award for Early Achievement in Research in 2007, and Litton Industries Professorship in 2007.