Modeling Network Coded TCP Throughput: A Simple Model and its Validation

MinJi Kim
MIT
Cambridge, MA 02139
minjikim@mit.edu

Muriel Médard
MIT
Cambridge, MA 02139
medard@mit.edu

João Barros
University of Porto
Porto, Portugal
jbarros@fe.up.pt

ABSTRACT

We analyze the performance of TCP and TCP with network coding (TCP/NC) in lossy wireless networks. We build upon the simple framework introduced by Padhye *et al.* and characterize the throughput behavior of classical TCP as well as TCP/NC as a function of erasure rate, round-trip time, maximum window size, and duration of the connection. Our analytical results show that network coding masks random erasures from TCP, thus preventing TCP's performance degradation in lossy networks (e.g. wireless networks). It is further seen that TCP/NC has significant throughput gains over TCP. Our analysis and simulation results show very close concordance and support that TCP/NC is robust against erasures. TCP/NC is not only able to increase its window size faster but also to maintain a large window size despite the random losses, whereas TCP experiences window closing because losses are mistakenly attributed to congestion. Note that network coding only masks random erasures, and allows TCP to react to congestion; thus, when there are correlated losses, TCP/NC also closes its window.

1. INTRODUCTION

The Transmission Control Protocol (TCP) is one of the core protocols of today's Internet Protocol Suite. TCP was designed for reliable transmission over wired networks, in which losses are generally indication of congestion. This is not the case in wireless networks, where losses are often due to fading, interference, and other physical phenomena. In wireless networks, TCP often incorrectly assumes that there is congestion within the network and unnecessarily reduces its transmission rate, when it should have actually transmitted continuously to overcome the lossy links. Consequently, TCP's performance in wireless networks is poor when compared to the wired counterparts as shown e.g. in [1,2]. There has been extensive research to combat these harmful effects of erasures and failures; however, TCP even with modifications does not achieve significant improvement. For example, there has been suggestions to allow TCP sender to maintain a large transmission window to overcome the random losses within the network. However, as we shall show in this paper, just keeping the window open does not lead to improvements in TCP's performance. Even if the transmission window is kept open, the sender can not transmit additional packets into the network without receiving acknowledgments. References [3, 4] give an overview and a comparison of various TCP versions over wireless links.

Some relief may come from network coding [5], which has been introduced as a potential paradigm to operate communication networks, in particular wireless networks. Network coding allows and encourages mixing of data at intermediate nodes, which has been shown to increase throughput and robustness against failures and erasures [6]. There are several practical protocols that take advantage of network coding in wireless networks [7–10].

In order to combine the benefits of TCP and network coding, [11] proposes a new protocol called TCP/NC. TCP/NC modifies TCP's acknowledgment (ACK) scheme such that it acknowledges *degrees of freedom* instead of individual packets, as shown in Figure 1. This is done so by using the concept of "seen" packets – in which the number of degrees of freedom received is translated to the number of consecutive packets received.

In this paper, we present a performance evaluation of TCP as well as TCP/NC in lossy networks. We adopt the same TCP model as in [2] – *i.e.* we consider standard TCP with Go-Back-N pipelining. Thus, the standard TCP discards packets that are out-of-order. We analytically show the throughput gains of TCP/NC over standard TCP, and present simulations results that support this analysis. We characterize the steady state throughput behavior of both TCP and TCP/NC as a function of erasure rate, round-trip time (RTT), and maximum window size. Our work thus extends the work of [2] for TCP and TCP/NC in lossy wireless networks. Furthermore, we use NS-2 (Network Simulator [12]) to verify our analytical results for TCP and TCP/NC. Our analysis and simulations show that TCP/NC is robust against erasures and failures. TCP/NC is not only able to increase its window size faster but also maintain a large window size despite losses within the network. Thus, TCP/NC is well suited for reliable communication in lossy networks. In contrast, standard TCP experiences window closing as losses are mistaken to be congestion.

There has been extensive research on modeling and analyzing TCP's performance [13–18]. Our goal is to present an analysis for TCP/NC, and to provide a comparison of TCP and TCP/NC in a lossy wireless environment. We adopt Padhye *et al.*'s model [2] as their model provides a simple yet good model to predict the performance of TCP. It would be interesting to extend and analyze TCP/NC in other TCP models in the literature.

The paper is organized as follows. In Section 2, we provide a brief overview of TCP/NC. In Section 3, we introduce our communication model. In Section 4, we provide the intuition behind the benefit of using network coding with TCP. Then, we provide throughput analysis for TCP and TCP/NC in Sections 5 and 6, respectively. In Section 7, we provide simulation results to verify our analytical results in Sections 5 and 6. Finally, we conclude in Section 8.

2. OVERVIEW OF TCP/NC

Reference [11] introduces a new *network coding* layer be-

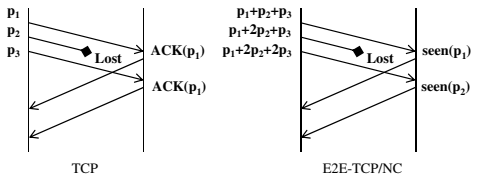


Figure 1: Example of TCP and TCP/NC. In the case of TCP, the TCP sender receives duplicate ACKs for packet $\mathbf{p_1}$, which may wrongly indicate congestion. However, for TCP/NC, the TCP sender receives ACKs for packets $\mathbf{p_1}$ and $\mathbf{p_2}$; thus, the TCP sender perceives a longer round-trip time (RTT) but does not mistake the loss to be congestion.

tween the TCP and IP in the protocol stack. The network coding layer intercepts and modifies TCP's acknowledgment (ACK) scheme such that random erasures does not affect the transport layer's performance. To do so, the *encoder*, the network coding unit under the sender TCP, transmits R random linear combinations of the buffered packets for every transmitted packet from TCP sender. The parameter R is the *redundancy factor*. Redundancy factor helps TCP/NC to recover from random losses; however, it cannot mask correlated losses, which are usually due to congestion. The *decoder*, the network coding unit under the receiver TCP, acknowledges *degrees of freedom* instead of individual packets, as shown in Figure 1. Once enough degrees of freedoms are received at the decoder, the decoder solves the set of linear equations to decode the original data transmitted by the TCP sender, and delivers the data to the TCP receiver.

We briefly note the overhead associated with network coding. The main overhead associated with network coding can be considered in two parts: 1) the coding vector (or coefficients) that has to be included in the header; 2) the encoding/decoding complexity. For receiver to decode a network coded packet, the packet needs to indicate the coding coefficients used to generate the linear combination of the original data packets. The overhead associated with the coefficients depend on the field size used for coding as well as the number of original packets combined. It has been shown that even a very small field size of $\mathbf{F}_{256}$ (*i.e.* 8 bits = 1 byte per coefficient) can provide a good performance [11, 19]. Therefore, even if we combine 50 original packets, the coding coefficients amount to 50 bytes over all. Note that a packet is typically around 1500 bytes. Therefore, the overhead associated with coding vector is not substantial. The second overhead associated with network coding is the encoding and decoding complexity, and the delay associated with the coding operations. Note that to affect TCP's performance, the decoding/encoding operations must take substantial amount of time to affect the round-trip time estimate of the TCP sender and receiver. However, we note that the delay caused the coding operations is negligible compared to the network round-trip time. For example, the network round-trip time is often in milliseconds (if not in hundreds of milliseconds), while a encoding/decoding operations involve a matrix multiplication/inversion in $\mathbf{F}_{256}$ which can be performed in a few microseconds.

In [11], the authors present two versions of TCP/NC – one that adheres to the end-to-end philosophy of TCP, in which coding operations are only performed at the source and destination; another that takes advantage of network coding even further by allowing any subset of intermediate nodes to re-encode. Note that re-encoding at the intermediate nodes is an optional feature, and is not required for TCP/NC to work. Here, we focus on TCP/NC with end-to-end network coding, which we denote E2E-TCP/NC (or in short E2E). However, a similar analysis applies to TCP/NC with re-encoding.

3. A MODEL FOR CONGESTION CONTROL

We focus on TCP's congestion avoidance mechanism, where the congestion control window size W is incremented by $1/W$ each time an ACK is received. Thus, when every packet in the congestion control window is ACKed, the window size W is increased to $W + 1$. On the other hand, the window size W is reduced whenever an erasure/congestion is detected.

We model TCP's behavior in terms of *rounds* [2]. We denote W_i to be the size of TCP's congestion control window size at the beginning of round i. The sender transmit W_i packets in its congestion window at the start of round i, and once all W_i packets have been sent, it defers transmitting any other packets until at least one ACK for the W_i packets are received. The ACK reception ends the current round, and starts round $i + 1$.

For simplicity, we assume that the duration of each round is equal to a round trip time (RTT), independent of W_i. This assumes that the time needed to transmit a packet is much smaller than the round trip time. This implies the following sequence of events for each round i: first, W_i packets are transmitted. Some packets may be lost. The receiver transmits ACKs for the received packets. (Note that TCP uses cumulative ACKs. Therefore, if the packets $1, 2, 3, 5, 6$ arrive at the receiver in sequence, then the receiver ACKs packets $1, 2, 3, 3, 3$. This signals that it has not yet received packet 4.) Some of the ACKs may also be lost. Once the sender receives the ACKs, it updates its window size. Assume that a_i packets are acknowledged in round i. Then, $W_{i+1} \leftarrow W_i + a_i/W_i$.

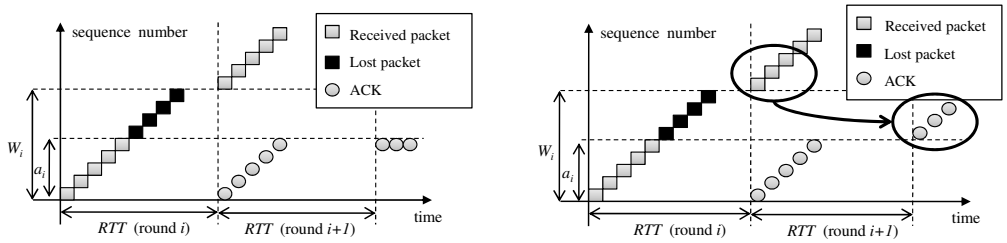
TCP reduces the window size for congestion control using the following two methods.

1) *Triple-duplicate (TD):* When the sender receives four ACKs with the same sequence number, then $W_{i+1} \leftarrow \frac{1}{2}W_i$.

2) *Time-out (TO):* If the sender does not hear from the receiver for a predefined time period, called the "time-out" period (which is T_o rounds long), then the sender closes its transmission window, $W_{i+1} \leftarrow 1$. At this point, the sender updates its TO period to $2T_o$ rounds, and transmits one packet. For any subsequent TO events, the sender transmits the one packet within its window, and doubles its TO period until $64T_o$ is reached, after which the TO period is fixed to $64T_o$. Once the sender receives an ACK from the receiver, it resets its TO period to T_o and increments its window according to the congestion avoidance mechanism. During time-out, the throughput of both TCP and E2E is zero.

Finally, in practice, the TCP receiver sends a single cumulative ACK after receiving β number of packets, where $\beta = 2$ typically. However, we assume that $\beta = 1$ for simplicity. Extending the analysis to $\beta \geq 1$ is straightforward.

There are several variants to the traditional TCP congestion control. For example, STCP [20] modifies the congestion control mechanism for networks with high bandwidth-delay products. Other variants include those with selective acknowledgment schemes [21]. It may be interesting to



(a) TCP (b) E2E-TCP/NC

Figure 2: The effect of erasures: TCP experiences triple-duplicate ACKs, and results in $W_{i+2} \leftarrow W_{i+1}/2$. However, E2E-TCP/NC masks the erasures using network coding, which allows TCP to advance its window. This figure depicts the sender's perspective, therefore, it indicates the time at which the sender transmits the packet or receives the ACK.

compare the performance of the TCP variants with that of TCP/NC. However, we focus on traditional TCP here.

3.1 Maximum window size

In general, TCP cannot increase its window size unboundedly; there is a maximum window size $W_{\max}$. The TCP sender uses a congestion avoidance mechanism to increment the window size until $W_{\max}$, at which the window size remains $W_{\max}$ until a TD or a TO event.

3.2 Erasures

We assume that there is are random erasures within in the network. We denote p to be the probability that a packet is lost at any given time. We assume that packet losses are independent. We note that this erasure model is different from that of [2] where losses are correlated within a round – *i.e.* bursty erasures. Correlated erasures model well bursty traffic and congestion in wireline networks. In our case, however, we are aiming to model wireless networks, thus we shall use random independent erasures.

We do not model congestion or correlated losses within this framework, but show by simulation that when there are correlated losses, both TCP and E2E close their window; thus, E2E is able to react to congestion.

3.3 Performance metric

We analyze the performance of TCP and E2E in terms of two metrics: the average throughput $\mathcal{T}$, and the expected window evolution $E[W]$, where $\mathcal{T}$ represents the total average throughput while window evolution $E[W]$ reflects the perceived throughput at a given time. We define $\mathcal{N}_{[t_1,t_2]}$ to be the number of packets received by the receiver during the interval $[t_1, t_2]$. The total average throughput is defined as:

$$\mathcal{T} = \lim_{\Delta \to \infty} \frac{\mathcal{N}_{[t,t+\Delta]}}{\Delta}. \quad (1)$$

We denote $\mathcal{T}_{tcp}$ and $\mathcal{T}_{e2e}$ to be the average throughput for TCP and E2E, respectively.

4. INTUITION

For traditional TCP, random erasures in the network can lead to triple-duplicate ACKs. For example, in Figure 2a, the sender transmits W_i packets in round i; however, only a_i of them arrive at the receiver. As a result, the receiver ACKs the a_i packets and waits for packet $a_i + 1$. When the sender receives the ACKs, round $i + 1$ starts. The sender updates its window ($W_{i+1} \leftarrow W_i + a_i/W_i$), and starts transmitting the new packets in the window. However, since the receiver is still waiting for packet $a_i + 1$, any other packets cause the receiver to request for packet $a_i + 1$. This results in a triple-duplicate ACKs event and the TCP sender closes its window, *i.e.* $W_{i+2} \leftarrow \frac{1}{2}W_{i+1} = \frac{1}{2}(W_i + a_i/W_i)$.

Notice that this window closing due to TD does not occur when using E2E as illustrated in Figure 2b. With network coding, any linearly independent packet delivers new information. Thus, any subsequent packet (in Figure 2b, the first packet sent in round $i + 1$) can be viewed as packet $a_i + 1$. As a result, the receiver is able to increment its ACK and the sender continues transmitting data. It follows that network coding masks the losses within the network from TCP, and prevents it from closing its window by misjudging link losses as congestion. ***Network coding translates random losses as longer RTT***, thus slowing down the transmission rate to adjust for losses without closing down the window in a drastic fashion.

Note that network coding does not mask correlated (or bursty) losses due to congestion. With enough correlated losses, network coding cannot correct for all the losses. As a result, the transmission rate will be adjusted according to standard TCP's congestion control mechanism when TCP/NC detects correlated losses. Therefore, network coding allows TCP to maintain a high throughput connection in a lossy environment; at the same time, allows TCP to react to congestion. Thus, network coding naturally distinguishes congestion from random losses for TCP.

5. ANALYSIS FOR TCP

We consider the effect of losses for TCP. The throughput analysis for TCP is similar to that of [2]. However, the model has been modified from that of [2] to account for independent losses and allow a fair comparison with network coded TCP. TCP can experience a TD or a TO event from random losses.

We note that, despite independent packet erasures, a single packet loss may affect subsequent packet reception. This is due to the fact that TCP requires in-order reception. A single packet loss within a transmission window forces all subsequent packets in the window to be out of order. Thus, they are discarded by the TCP receiver. As a result, standard TCP's throughput behavior with independent losses is similar to that of [2], where losses are correlated within one round.

5.1 Triple-duplicate for TCP

We consider the expected throughput between consecutive TD events, as shown in Figure 3. Assume that the TD events occurred at time t_1 and $t_2 = t_1 + \Delta$, $\Delta > 0$. Assume that

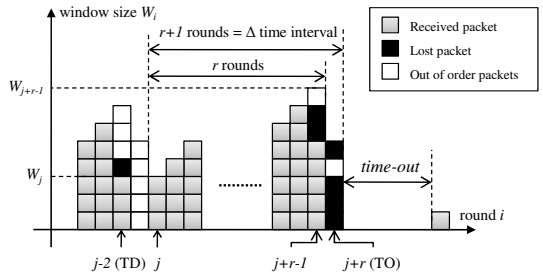


Figure 3: TCP's window size with a TD event and a TO event. In round $j-2$, losses occur resulting in triple-duplicate ACKs. On the other hand, in round $j+r-1$, losses occur; however, in the following round $j+r$ losses occur such that the TCP sender only receives two-duplicate ACKs. As a result, TCP experiences time-out.

round j begins immediately after time t_1, and that packet loss occurs in the r-th round, *i.e.* round $j+r-1$.

First, we calculate $E[\mathcal{N}_{[t_1,t_2]}]$. Note that during the interval $[t_1, t_2]$, there are no packet losses. Given that the probability of a packet loss is p, the expected number of consecutive packets that are successfully sent from sender to receiver is

$$E\left[\mathcal{N}_{[t_1,t_2]}\right] = \left(\sum_{k=1}^{\infty} k(1-p)^{k-1}p\right) - 1 = \frac{1-p}{p}. \quad (2)$$

The packets (in white in Figure 3) sent after the lost packets (in black in Figure 3) are out of order, and will not be accepted by the standard TCP receiver. Thus, Equation (2) does not take into account the packets sent in round $j-1$ or $j+r$.

We calculate the expected time period between two TD events, $E[\Delta]$. As in Figure 3, after the packet losses in round j, there is an additional round for the loss feedback from the receiver to reach the sender. Therefore, there are $r+1$ rounds within the time interval $[t_1, t_2]$, and $\Delta = RTT(r+1)$. Thus,

$$E[\Delta] = RTT(E[r]+1). \quad (3)$$

To derive $E[r]$, note that $W_{j+r-1} = W_j + r - 1$ and

$$W_j = \frac{1}{2}W_{j-1} = \frac{1}{2}\left(W_{j-2} + \frac{a_{j-2}}{W_{j-2}}\right). \quad (4)$$

Equation (4) is due to TCP's congestion control. TCP interprets the losses in round $j-2$ as congestion, and as a result halves its window. Assuming that, in the long run, $E[W_{j+r-1}] = E[W_{j-2}]$ and that a_{j-2} is uniformly distributed between $[0, W_{j-2}]$,

$$E[W_{j+r-1}] = 2\left(E[r] - \frac{3}{4}\right) \text{ and } E[W_j] = E[r] - \frac{1}{2}. \quad (5)$$

During these r rounds, we expect to successfully transmit $\frac{1-p}{p}$ packets as noted in Equation (2). This results in:

$$\frac{1-p}{p} = \left(\sum_{k=0}^{r-2} W_{j+k}\right) + a_{j+r-1} \quad (6)$$

$$= (r-1)W_j + \frac{(r-1)(r-2)}{2} + a_{j+r-1}. \quad (7)$$

Taking the expectation of Equation (7) and using Equation (5),

$$\frac{1-p}{p} = \frac{3}{2}(E[r]-1)^2 + E[a_{j+r-1}]. \quad (8)$$

Note that a_{j+r-1} is assumed to be uniformly distributed across $[0, W_{j+r-1}]$. Thus, $E[a_{j+r-1}] = E[W_{j+r-1}]/2 = E[r] - \frac{3}{4}$ by Equation (5). Solving Equation (8) for $E[r]$, we find:

$$E[r] = \frac{2}{3} + \sqrt{-\frac{1}{18} + \frac{2}{3}\frac{1-p}{p}}. \quad (9)$$

The steady state $E[W]$ is the average window size over two consecutive TD events. This provides an expression of steady state average window size for TCP (using Equations (5)):

$$E[W] = \frac{E[W_j] + E[W_{j+r-1}]}{2} = \frac{3}{2}E[r] - 1. \quad (10)$$

The average throughput can be expressed as

$$\mathcal{T}'_{tcp} = \frac{E[\mathcal{N}_{[t_1,t_2]}]}{E[\Delta]} = \frac{1-p}{p}\frac{1}{RTT(E[r]+1)}. \quad (11)$$

For small p, $\mathcal{T}'_{tcp} \approx \frac{1}{RTT}\sqrt{\frac{3}{2p}} + o(\frac{1}{\sqrt{p}})$; for large p, $\mathcal{T}'_{tcp} \approx \frac{1}{RTT}\frac{1-p}{p}$. If we only consider TD events, the long-term steady state throughput is equal to that in Equation (11).

The analysis above assumes that the window size can grow unboundedly; however, this is not the case. To take maximum window size $W_{\max}$ into account, we make a following approximation:

$$\mathcal{T}_{tcp} = \min\left(\frac{W_{\max}}{RTT}, \mathcal{T}'_{tcp}\right). \quad (12)$$

For small p, this result coincide with the results in [2].

5.2 Time-out for TCP

If there are enough losses within two consecutive rounds, TCP may experience a TO event, as shown in Figure 3. Thus, $\mathbf{P}(\text{TO}|W)$, the probability of a TO event given a window size of W, is given by

$$\mathbf{P}(\text{TO}|W) = \begin{cases} 1 & \text{if } W < 3; \\ \sum_{i=0}^{2}\binom{W}{i}p^{W-i}(1-p)^i & \text{if } W \geq 3. \end{cases} \quad (13)$$

Note that when the window is small ($W < 3$), then losses result in TO events. For example, assume $W = 2$ with packets $\mathbf{p_1}$ and $\mathbf{p_2}$ in its window. Assume that $\mathbf{p_2}$ is lost. Then, the TCP sender may send another packet $\mathbf{p_3}$ in the subsequent round since the acknowledgment for $\mathbf{p_1}$ allows it to transmit a new packet. However, this would generate a single duplicate ACK with no further packets in the pipeline, and TCP sender waits for ACKs until it times out.

We approximate W in above Equation (13) with the expected window size $E[W]$ from Equation (10). The length of the TO event depends on the duration of the loss events. Thus, the expected duration of TO period (in RTTs) is given in Equation (15). Finally, by combining the results in Equations (12), (13), and (15), we get an expression for the average throughput of TCP as shown in Equation (16).

6. ANALYSIS FOR E2E-TCP/NC

We consider the expected throughput for E2E. Note that erasure patterns that result in TD and/or TO events under TCP may not yield the same result under E2E, as illustrated

$$E[\text{duration of TO period}] = (1-p)\left[T_o p + 3T_o p^2 + 7T_o p^3 + 15T_o p^4 + 31T_o p^5 + \sum_{i=0}^{\infty}(63 + i\cdot 64)T_o p^{6+i}\right] \quad (14)$$

$$= (1-p)\left[T_o p + 3T_o p^2 + 7T_o p^3 + 15T_o p^4 + 31T_o p^5 + 63T_o \frac{p^6}{1-p} + 64T_o \frac{p^7}{(1-p)^2}\right] \quad (15)$$

$$\mathcal{T}_{tcp} = \min\left(\frac{W_{\max}}{RTT}, \frac{1-p}{p}\frac{1}{RTT\left(\frac{5}{3} + \sqrt{-\frac{1}{18} + \frac{2}{3}\frac{1-p}{p}} + \mathbf{P}(\text{TO}|E[W])E[\text{duration of TO period}]\right)}\right) \quad (16)$$

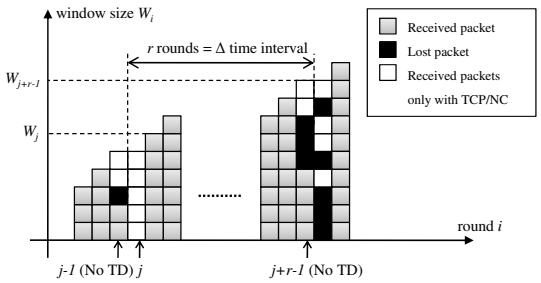


Figure 4: E2E-TCP/NC's window size with erasures that would lead to a triple-duplicate ACKs event when using standard TCP. Note that unlike TCP, the window size is non-decreasing.

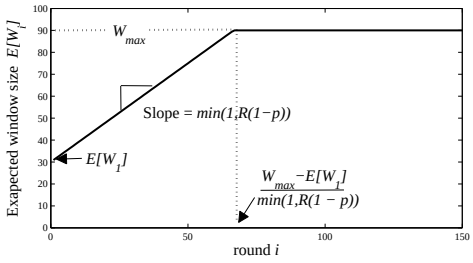


Figure 5: Expected window size for E2E where $W_{\max} = 90$, $E[W_1] = 30$. We usually assume $E[W_1] = 1$; here we use $E[W_1] = 30$ to exemplify the effect of $E[W_1]$.

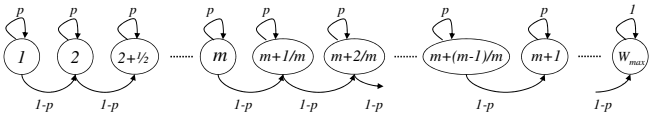


Figure 6: Markov chain for the E2E's window evolution.

$$P = \begin{pmatrix} p & 1-p & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & p & 1-p & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & p & 1-p & 0 & \cdots & 0 & 0 \\ \vdots & & & \ddots & \ddots & \ddots & & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & p & 1-p \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 7: The *transition matrix* P for the Markov chain in Figure 6. The shaded part of the matrix is denoted Q. Matrix $N = (I - Q)^{-1}$ is the *fundamental matrix* of the Markov chain, and can be used to compute the expected rounds until the absorption state.

in Section 4. We emphasize again that this is due to the fact that any linearly independent packet conveys a new degree of freedom to the receiver. Figure 4 illustrates this effect – packets (in white) sent after the lost packets (in black) are acknowledged by the receivers, thus allowing E2E to advance its window. This implies that E2E does not experience window closing owing to random losses often.

6.1 E2E-TCP/NC Window Evolution

From Figure 4, we observe that E2E is able to maintain its window size despite experiencing losses. This is partially because E2E is able to receive packets that would be considered out of order by TCP. As a result, E2E's window evolves differently from that of TCP, and can be characterized by a simple recursive relationship as

$$E[W_i] = E[W_{i-1}] + \frac{E[a_{i-1}]}{E[W_{i-1}]} = E[W_{i-1}] + \min\{1, R(1-p)\}.$$

The recursive relationship captures the fact that every packet that is linearly independent of previously received packets is considered to be *innovative* and is therefore acknowledged. Consequently, any arrival at the receiver is acknowledged with high probability; thus, we expect $E[a_{i-1}]$ packets to be acknowledged and the window to be incremented by $\frac{E[a_{i-1}]}{E[W_{i-1}]}$. Note that $E[a_{i-1}] = (1-p)\cdot R \cdot E[W_{i-1}]$ since the encoder transmits on average R linear combinations for every packet transmitted by the TCP sender.

Once we take $W_{\max}$ into account, we have the following expression for E2E's expected window size:

$$E[W_i] = \min(W_{\max}, E[W_1] + i\min\{1, R(1-p)\}), \quad (17)$$

where i is the round number. $E[W_1]$ is the initial window size, and we set $E[W_1] = 1$. Figure 5 shows an example of the evolution of the E2E window using Equation (17).

6.1.1 Markov Chain Model

The above analysis describes the expected behavior of E2E's window size. We can also describe the window size behavior using a Markov chain as shown in Figure 6. The states of this Markov chain represent the instantaneous window size (not specific to a round). A transition occurs whenever a packet is transmitted. We denote $S(W)$ to be the state representing the window size of W. Assume that we are at state $S(W)$. If a transmitted packet is received by the E2E receiver and acknowledged, the window is incremented by $\frac{1}{W}$; thus, we end up in state $S(W + \frac{1}{W})$. This occurs with probability $(1-p)$. On the other hand, if the packet is lost, then we stay at $S(W)$. This occurs with probability p. Thus, the Markov chain states represent the window size, and the transitions correspond to packet transmissions.

Note that $S(W_{\max})$ is an absorbing state of the Markov chain. As noted in Section 4, E2E does not often experience

a window shutdown, which implies that there are correlated or heavy losses. Thus, E2E's window size is non-decreasing, as shown in Figure 6. Therefore, given enough time, E2E reaches state $S(W_{\max})$ with probability equal to 1. We analyze the expected number of packet transmissions needed for absorption.

The *transition matrix* P and the *fundamental matrix* $N = (I-Q)^{-1}$ of the Markov chain is given in Figure 7. The entry $N(S_1, S_2)$ represents the expected number of visits to state S_2 before absorption – *i.e.* we reach state $S(W_{\max})$ – when we start from state S_1. Our objective is to find the expected number of packets transmitted to reach $S(W_{\max})$ starting from state $S(E[W_1])$ where $E[W_1] = 1$. The partial sum of the first row entries of N gives the expected number of packets transmitted until we reach the window size W. The expression for the first row of N can be derived using cofactors: $N(1,:) = \left[\frac{1}{1-p}, \frac{1}{1-p}, \cdots, \frac{1}{1-p}\right]$. The expected number of packet transmissions $T_p(W)$ to reach a window size of $W \in [1, W_{\max}]$ is:

$$\begin{aligned} T_p(W) &= \sum_{m=S(1)}^{S(W)} N(1,m) = \sum_{m=S(1)}^{S(W)} \frac{1}{1-p} = \frac{1}{1-p} \sum_{m=S(1)}^{S(W)} 1 \\ &= \frac{\text{Number of states between } S(1) \text{ and } S(W)}{1-p} \\ &= \frac{W(W-1)}{2(1-p)}. \end{aligned} \tag{18}$$

$T_p(W)$ is the number of packets we expect to transmit given the erasure probability p. If we set $p = 0$, then $T_0(W) = \frac{W(W-1)}{2}$. Therefore, $\frac{W(W-1)}{2}$ is the minimal number of transmission needed to achieve W (since this assumes no packets are lost). Note that $\frac{T_p(W)}{T_0(W)} = \frac{1}{1-p}$ represents a lower bound on cost when losses are introduced – *i.e.* to combat random erasures, the sender on average has to send at least $\frac{1}{1-p}$ packets for each packet it wishes to send. This is exactly the definition of redundancy factor R. This analysis indicates that we should set $R \geq \frac{T_p(W)}{T_0(W)}$. Furthermore, $T_0(W)$ is equal to the area under the curve for rounds $i \in [0, \frac{W - E[W_1]}{\min\{1, R\cdot(1-p)\}}]$ in Figure 5 if we set $R \geq \frac{1}{1-p}$. A more detailed discussion on the effect of R is in Section 6.2.1.

6.2 E2E-TCP/NC Analysis per Round

Using the results in Section 6.1, we derive an expression for the throughput. The throughput of round i, $\mathcal{T}_i$, is directly proportional to the window size $E[W_i]$, *i.e.*

$$\mathcal{T}_i = \frac{E[W_i]}{SRTT} \min\{1, R(1-p)\} \text{ packets per second}, \tag{19}$$

where $SRTT$ is the round trip time estimate. The RTT and its estimate $SRTT$ play an important role in E2E. We shall formally define and discuss the effect of R and $SRTT$ below.

We note that $\mathcal{T}_i \propto (1-p) \cdot R \cdot E[W_i]$. At any given round i, E2E sender transmits $R \cdot E[W_i]$ coded packets, and we expect $pR \cdot E[W_i]$ packets to be lost. Thus, the E2E receiver only receives $(1-p) \cdot R \cdot E[W_i]$ degrees of freedom.

6.2.1 Redundancy Factor R

The redundancy factor $R \geq 1$ is the ratio between the average rate at which linear combinations are sent to the receiver and the rate at which TCP's window progresses. For example, if the TCP sender has 10 packets in its window, then the encoder transmits $10R$ linear combinations. If R is large enough, the receiver will receive at least 10 linear combinations to decode the original 10 packets. This redundancy is necessary to (a) compensate for the losses within the network, and (b) match TCP's sending rate to the rate at which data is actually received at the receiver. References [11, 19] introduce the redundancy factor with TCP/NC, and show that $R \geq \frac{1}{1-p}$ is necessary. This coincides with our analysis in Section 6.1.1.

The redundancy factor R should be chosen with some care. If $R < \frac{1}{1-p}$ causes significant performance degradation, since network coding can no longer fully compensate for the losses which may lead to window closing for E2E. To maximize throughput, an optimal value of $R \geq \frac{1}{1-p}$ should be chosen. However, setting $R \gg \frac{1}{1-p}$ may over-compensate for the losses within the network; thus, introducing more redundant packets than necessary. On the other hand, matching R to exactly $\frac{1}{1-p}$ may not be desirable for two reasons: 1) The exact value of $\frac{1}{1-p}$ may not be available or difficult to obtain in real applications; 2) As $R \to \frac{1}{1-p}$, it becomes more likely that E2E is unable to *fully* recover from losses in any given round. By *fully* recover, we mean that E2E decoder is able to acknowledge all packet transmitted in that round. As we shall show in Section 7, E2E can maintain a fairly high throughput with just partial acknowledgment (in each round, only a subset of the packets are acknowledged owing to losses). However, we still witness a degradation in throughput as R decreases. Thus, we assume that $R \geq \frac{1}{1-p}$.

6.2.2 Effective Round Trip Time $SRTT$

$SRTT$ is the round trip time estimate that TCP maintains by sampling the behavior of packets sent over the connection. It is denoted $SRTT$ because it is often referred to as "smoothed" round trip time as it is obtained by averaging the time for a packet to be acknowledged after the packet has been sent. We note that, in Equation (19), we use $SRTT$ instead of RTT because $SRTT$ is the "effective" round trip time E2E experiences.

In lossy networks, E2E's $SRTT$ is often greater than RTT. This can be seen in Figure 1. The first coded packet ($\mathbf{p_1} + \mathbf{p_2} + \mathbf{p_3}$) is received and acknowledged ($\mathbf{seen}(\mathbf{p_1})$). Thus, the sender is able to estimate the round trip time correctly; resulting in $SRTT = RTT$. However, the second packet ($\mathbf{p_1} + \mathbf{2p_2} + \mathbf{p_3}$) is lost. As a result, the third packet ($\mathbf{p_1} + \mathbf{2p_2} + \mathbf{2p_3}$) is used to acknowledge the second degree of freedom ($\mathbf{seen}(\mathbf{p_2})$). In our model, we assume for simplicity that the time needed to transmit a packet is much smaller than RTT; thus, despite the losses, our model would result in $SRTT \approx RTT$. However, in practice, depending on the size of the packets, the transmission time may not be negligible.

6.3 E2E-TCP/NC Average Throughput

Taking Equation (19), we can average the throughput over n rounds to obtain the average throughput for E2E-TCP/NC.

$$\begin{aligned} \mathcal{T}_{e2e} &= \frac{1}{n} \sum_{i=1}^{n} \frac{E[W_i]}{SRTT} \min\{1, R(1-p)\} \\ &= \frac{\sum_{i=1}^{n} \min(W_{\max}, E[W_1] + i)}{n \cdot SRTT} \text{ since } R \geq \frac{1}{1-p} \\ &= \frac{1}{n \cdot SRTT} \cdot f(n), \end{aligned} \tag{20}$$

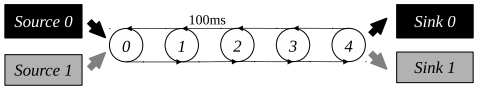


Figure 8: Network topology for the simulations.

where

$$f(n) = \begin{cases} nE[W_1] + \frac{n(n+1)}{2} & \text{for } n \leq r^* \\ nW_{\max} - r^*(W_{\max} - E[W_1]) + \frac{r^*(r^*-1)}{2} & \text{for } n > r^* \end{cases}$$

$r^* = W_{\max} - E[W_1]$.

Note that as $n \to \infty$, the average throughput $\mathcal{T}_{e2e} \to \frac{W_{\max}}{SRTT}$.

An important aspect of TCP is congestion control mechanism. This analysis may suggest that network coding no longer allows for TCP to react to congestion. We emphasize that the above analysis assumes that there are only random losses with probability p, and that there are no correlated losses. It is important to note that the erasure correcting power of network coding is limited by the redundancy factor R. If there are enough losses (e.g., losses caused by congestion), network coding cannot mask all the erasures from TCP. This may lead E2E to experience a TD or TO event, depending on the variants of TCP used. In Section 7.3, we present simulation results that show that TCP's congestion control mechanism still applies to E2E when appropriate.

7. SIMULATION RESULTS

We use simulations to verify that our analysis captures the behavior of both TCP and E2E. We use NS-2 (Network Simulator [12]) to simulate TCP and E2E-TCP/NC, where we use the implementation of E2E from [19]. Two FTP applications (ftp0, ftp1) wish to communicate from the source (src0, src1) to sink (sink0, sink1), respectively. There is no limit to the file size. The sources generate packets continuously until the end of the simulation. The two FTP applications use either TCP or E2E-TCP/NC. We denote TCP0, TCP1 to be the two FTP applications when using TCP; and we denote NC0, NC1 to be the two FTP applications when using E2E.

The network topology for the simulation is shown in Figure 8. All links, in both forward and backward paths, are assumed to have a bandwidth of C Mbps, a propagation delay of 100 ms, a buffer size of 200, and a erasure rate of q. Note that since there are in total four links in the path from node 0 to node 4, the probability of packet erasure is $p = 1 - (1-q)^4$. Each packet transmitted is assumed to be 8000 bits (1000 bytes). We set $W_{\max} = 50$ packets for all simulations. In addition, time-out period $T_o = \frac{3}{RTT} = 3.75$ rounds long (3 seconds). Therefore, our variables for the simulations are:

- $p = 1 - (1-q)^4$: End-to-end erasure rate,
- R: Redundancy factor,
- C: Capacity of the links (in Mbps).

We study the effect these variables have on the following:

- $\mathcal{T}$: Throughput of TCP or E2E,
- $E[W]$: Average window size of TCP or E2E,
- $SRTT$: Round-trip estimate.

For each data point, we average the performance over 100 independent runs of the simulation, each of which is 1000 seconds long.

7.1 Probability of erasure p

We set $C = 2$ Mbps and $R = 1.25$ regardless of the value of p. We vary q to be 0, 0.005, 0.015, 0.025, and 0.05. The corresponding p values are 0, 0.0199, 0.0587, 0.0963, and 0.1855. The results are shown in Figures 9, 10, and 11.

Firstly, we show that when there are no random erasures ($p = 0$), then E2E and TCP behave similarly, as shown in Figures 9a, 10a, and 11a. Without any random losses and congestion, all of the flows (NC0, NC1, TCP0, TCP1) achieve the maximal throughput, $\frac{W_{\max}}{RTT} \cdot \frac{8}{10^6} = 0.5$ Mbps.

The more interesting result is when $p > 0$. As our analysis predicts, E2E sustains its high throughput despite the random erasures in the network. We observe that TCP may close its window due to triple-duplicates ACKs or time-outs; however, E2E is more resilient to such erasure patterns. Therefore, E2E is able to increment its window consistently, and *maintain* the window size of 50 even under lossy conditions when standard TCP is unable to (resulting in the window fluctuation in Figure 10).

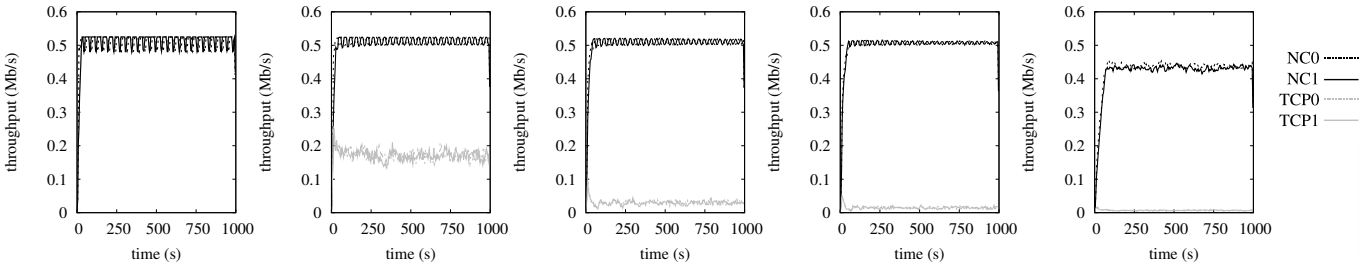
An interesting observation is that, TCP achieves a moderate average window size although the throughput (Mbps) is much lower (Figures 9 and 10). This shows that naïvely keeping the transmission window open is not sufficient to overcome the random losses within the network, and does not lead to improvements in TCP's performance. Even if the transmission window is kept open (e.g. during timeout period), the sender can not transmit additional packets into the network without receiving ACKs. Eventually, this leads to a TD or TO event.

As described in Sections 4 and 6.2.2, E2E masks errors by translating losses as longer RTT. For E2E, if a specific packet is lost, the next subsequent packet received can "replace" the lost packet; thus, allowing the receiver to send an ACK. Therefore, the longer RTT estimate takes into account the delay associated with waiting for the next subsequent packet at the receiver. In Figure 11, we verify that this is indeed true. TCP, depending on the ACKs received, modifies its RTT estimation; thus, due to random erasures, TCP's RTT estimate fluctuates significantly. On the other hand, E2E is able to maintain a consistent estimate of the RTT; however, is slightly above the actual 800 ms.

7.2 Redundancy factor R

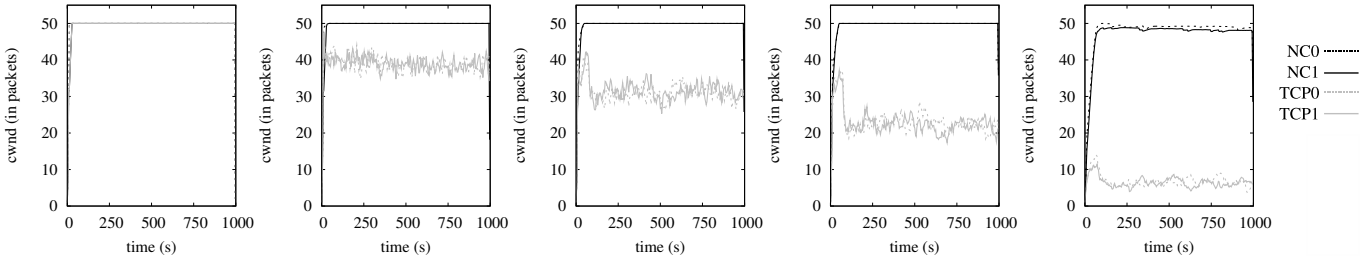
We set $C = 2$ Mbps. We vary the value of p and R to understand the relationship between R and p. In Section 6.2.1, we noted that $R \geq \frac{1}{1-p}$ is necessary to mask random erasures from TCP. However, as $R \to \frac{1}{1-p}$, the probability that the erasures are completely masked decreases. This may suggest that we need $R \gg \frac{1}{1-p}$ for E2E to maintain its high throughput. However, we shall show that R need not be much larger than $\frac{1}{1-p}$ for E2E to achieve its maximal throughput.

In Figure 12, we present E2E throughput behavior with $p = 0.0963$ and varying R. Note that $\frac{1}{1-p} = 1.107$ for $p = 0.0963$. There is a dramatic change in throughput behavior as we increase R from 1.11 to 1.12. Note that $R = 1.12$ is only 1% additional redundancy than the theoretical minimum, *i.e.* $\frac{1.12}{1/(1-p)} \approx 1.01$. Another interesting observation is that, even with $R = 1.10$ or $R = 1.11$, E2E



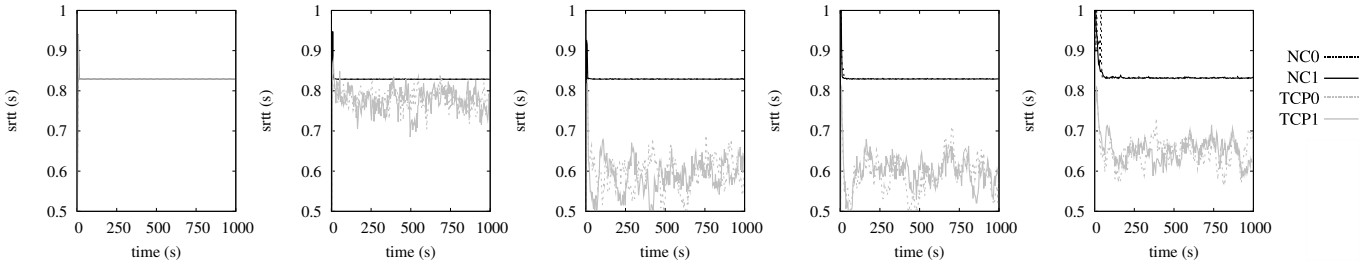
(a) $p = 0$ (b) $p = 0.0199$ (c) $p = 0.0587$ (d) $p = 0.0963$ (e) $p = 0.1855$

Figure 9: Throughput of E2E-TCP/NC and TCP with varying link erasure probability p.



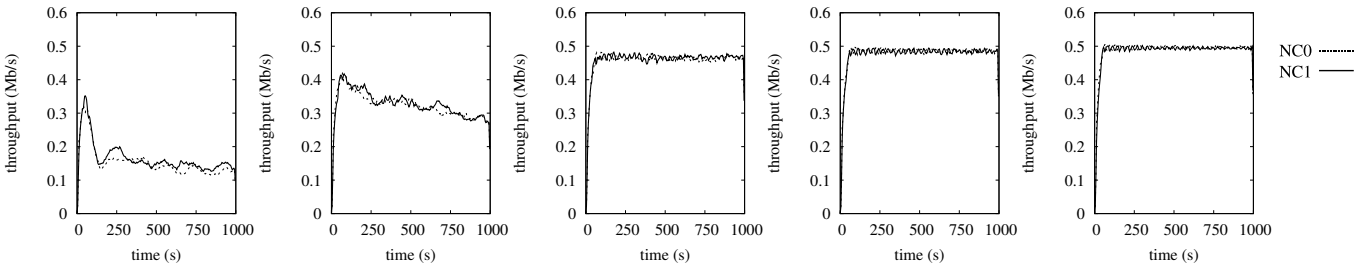
(a) $p = 0$ (b) $p = 0.0199$ (c) $p = 0.0587$ (d) $p = 0.0963$ (e) $p = 0.1855$

Figure 10: The congestion window size of E2E-TCP/NC and TCP with varying link erasure probability p.



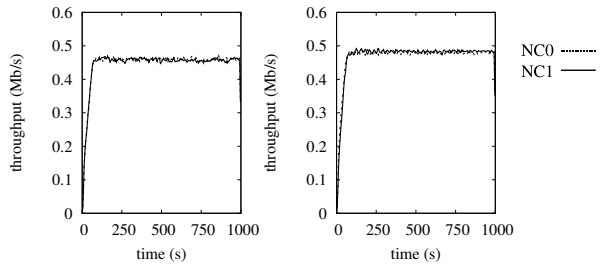
(a) $p = 0$ (b) $p = 0.0199$ (c) $p = 0.0587$ (d) $p = 0.0963$ (e) $p = 0.1855$

Figure 11: The round trip time estimate (SRTT) of E2E-TCP/NC and TCP with varying link erasure probability p.



(a) $R = 1.10$ (b) $R = 1.11$ (c) $R = 1.12$ (d) $R = 1.13$ (e) $R = 1.15$

Figure 12: Throughput of E2E-TCP/NC for $p = 0.0963$ with varying redundancy factor R. Note that $\frac{1}{1-p} = 1.107$.



(a) $R = 1.26$ (b) $R = 1.28$

Figure 13: Throughput of E2E-TCP/NC for $p = 0.1855$ with varying redundancy factor R. Note that $\frac{1}{1-p} = 1.228$.

achieves a significantly higher throughput than TCP (in Figure 9d) for $p = 0.0963$.

Figure 9e shows that, with $p = 0.1855$, E2E throughput is not as steady, and does not achieve the maximal throughput of 0.5 Mbps. This is because $\frac{1}{1-p} = 1.23$ is very close to $R = 1.25$. As a result, $R = 1.25$ is not sufficient to mask erasures with high probability. In Figure 13, we show that E2E achieves an average throughput of 0.5 Mbps once $R \geq 1.28$. Note that $R = 1.28$ is only 4% additional redundancy than the theoretical minimum, *i.e.* $\frac{1.28}{1/(1-p)} \approx 1.04$.

Similar behavior can be observed for $p = 0.0199$ and 0.0587, and setting R to be slightly above $\frac{1}{1-p}$ is sufficient. A good heuristic to use in setting R is the following. Given a probability of erasure p and window size W, the probability that losses in any given round is completely masked is upper bounded by $\sum_{x=0}^{W(R-1)} \binom{RW}{x} p^x (1-p)^{RW-x}$, *i.e.* there are no more than $W(R-1)$ losses in a round. Ensuring that this probability is at least 0.8 has proven to be a good heuristic to use in finding the appropriate value of R.

7.3 Congestion Control

We showed that E2E achieves a good performance in lossy environment. This may raise concerns about masking correlated losses from TCP; thus, disabling TCP's congestion control mechanism. We show that the network coding layer masks random losses only, and allows TCP's congestion control to take affect when necessary.

Given a capacity C and erasure rate p, the available bandwidth is $C(1-p)$ Mbps. Therefore, given two flows, a fair allocation of bandwidth should be $\frac{C(1-p)}{2}$ Mbps per flow. Note that this is the *available* bandwidth, not the *achieved* bandwidth. As we have seen, if $p > 0$, TCP may not be able to use fully the available bandwidth. On the other hand, E2E is able to use the available bandwidth efficiently. With E2E flows, there is another parameter we need to consider: the redundancy factor R. Since E2E sends R coded packets for each data packet, the achievable bandwidth is $\min\{C(1-p), \frac{C}{R}\}$ Mbps; if shared among two flows fairly, we expect $\frac{1}{2}\min\{C(1-p), \frac{C}{R}\}$ Mbps per coded flow. Note that, if R is chosen appropriately (*i.e.* slightly above $\frac{1}{1-p}$), then E2E can achieve rate close to $C(1-p)$, which is optimal.

We show that multiple E2E flows share the bandwidth fairly. We consider two flows (NC0, NC1) with $W_{\max} = 50$, $R = 1.2$, and $p = 0.0963$. If there is no congestion, each flow would achieve approximately 0.5 Mbps. However, we set $C = 0.7$ Mbps. The two flows should achieve $\frac{1}{2}\min\{0.7(1-0.0963), \frac{0.7}{1.2}\} = 0.2917$ Mbps. We observe in Figure 14 that NC0 and NC1 achieve 0.2878 Mbps and 0.2868 Mbps, respectively. Note that $\frac{C(1-p)}{2} = 0.3162$; thus, NC0 and NC1 is near optimal even though $R = 1.2 > \frac{1}{1-p} = 1.106$.

For our next simulations, we set $C = 0.9$ Mbps, $W_{\max} = 50$, $p = 0.0963$, and $R = 1.2$. Furthermore, we assume that NC0 starts at 0s, and runs for 1000s, while NC1 starts at time 350s and ends at time 650s. Before NC1 enters, NC0 should be able to achieve a throughput of 0.5 Mbps; however, when NC1 starts its connection, there is congestion, and both NC0 and NC1 have to react to this. Figure 15 shows that indeed this is true. We observe that when NC1 starts its connection, both NC0 and NC1 shares the bandwidth equally (0.3700 and 0.3669 Mbps, respectively). The achievable bandwidth predicted by $\min\{C(1-p), \frac{C}{R}\}$ is 0.75 Mbps (or 0.375 Mbps per flow). Note that both NC0 and NC1 maintains its maximum window size of 50. Instead, NC0 and NC1 experience a longer RTT, which naturally translates to a lower throughput given the same $W_{\max}$.

7.4 Comparison to the analytical model

Finally, we examine the accuracy of our analytical model in predicting the behavior of TCP and E2E. First, note that our analytical model of window evolution (shown in Equation (17) and Figure 5) demonstrates the same trend as that of the window evolution of E2E NS-2 simulations (shown in Figure 10). Second, we compare the actual NS-2 simulation performance to the analytical model. This is shown in Table 1. We observe that Equations (19) and (17) predict well the trend of E2E's throughput and window evolution, and provides a good estimate of E2E's performance. Furthermore, our analysis predicts the average TCP behavior well. In Table 1, we see that Equation (16) is consistent with the NS-2 simulation results even for large values of p. Therefore, both simulations as well as analysis support that E2E is resilient to erasures; thus, better suited for reliable transmission over unreliable networks, such as wireless networks.

8. CONCLUSIONS

We have presented an analytical study and compared the performance of TCP and E2E-TCP/NC. Our analysis characterizes the throughput of TCP and E2E as a function of erasure rate, round-trip time, maximum window size, and the duration of the connection. We showed that network coding, which is robust against erasures and failures, can prevent TCP's performance degradation often observed in lossy networks. Our analytical model shows that TCP with network coding has significant throughput gains over TCP. E2E is not only able to increase its window size faster but also to maintain a large window size despite losses within the network; on the other hand, TCP experiences window closing as losses are mistaken to be congestion. Furthermore, NS-2 simulations verify our analysis on TCP's and E2E's performance. Our analysis and simulation results both support that E2E is robust against erasures and failures. Thus, E2E is well suited for reliable communication in lossy wireless networks.

Acknowledgments: This work is supported by the MIT-Portugal Program under award No: 014098-153.

9. REFERENCES

[1] R. Cáceres and L. Iftode, "Improving the performance of reliable transport protocols in mobile computing

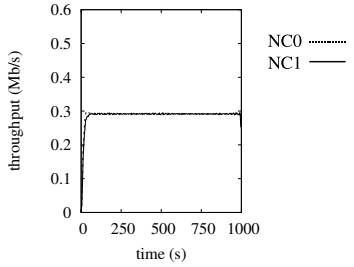


Figure 14: E2E-TCP/NC for $p = 0.0963$ and $C = 0.7$ Mbps.

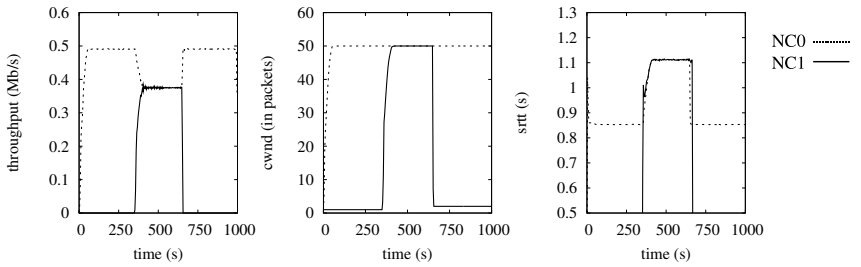


Figure 15: E2E-TCP/NC for $p = 0.0963$ with congestion ($C = 0.9$ Mbps, $R = 1.2$, $W_{\max} = 50$).

Table 1: The average simulated or predicted long-term throughput of TCP and E2E in megabits per second (Mbps). 'NC0', 'NC1', 'TCP0', 'TCP1' are average throughput achieved in the NS-2 simulations (with the corresponding 'R'). 'E2E analysis' is calculated using Equation (20) with $\lfloor n \cdot SRTT \rfloor = 1000$. 'TCP analysis' is computed using Equation (16).

p	E2E $SRTT$	R	NC0	NC1	E2E analysis	TCP0	TCP1	TCP analysis
0	0.8256	1	0.5080	0.5057	0.4819	0.5080	0.5057	0.5000
0.0199	0.8260	1.03	0.4952	0.4932	0.4817	0.1716	0.1711	0.0667
0.0587	0.8264	1.09	0.4926	0.4909	0.4814	0.0297	0.0298	0.0325
0.0963	0.8281	1.13	0.4758	0.4738	0.4804	0.0149	0.0149	0.0220
0.1855	0.8347	1.29	0.4716	0.4782	0.4766	0.0070	0.0070	0.0098

environments," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 5, June 1995.

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

[3] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving tcp performance over wireless links," *IEEE/ACM Transactions on Networking*, vol. 5, December 1997.

[4] Y. Tian, K. Xu, and N. Ansari, "TCP in wireless environments: Problems and solutions," *IEEE Comm. Magazine*, vol. 43, pp. 27–32, 2005.

[5] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, pp. 1204–1216, 2000.

[6] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transaction on Networking*, vol. 11, pp. 782–795, 2003.

[7] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: Practical wireless network coding," in *Proceedings of ACM SIGCOMM*, 2006.

[8] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *Proceedings of ACM SIGCOMM*, 2007.

[9] B. L. Yunfeng Lin, Baochun Li, "CodeOr: Opportunisitic routing in wireless mesh networks with segmented network coding," in *Proceedings of IEEE International Conference on Network Protocols*, 2008.

[10] J. Barros, R. A. Costa, D. Munaretto, and J. Widmer, "Effective delay control for online network coding," in *Proceedings of IEEE INFOCOM*, 2009.

[11] J. K. Sundararajan, D. Shah, M. Médard, M. Mitzenmacher, and J. Barros, "Network coding meets TCP," in *Proceedings of IEEE INFOCOM*, April 2009, pp. 280–288.

[12] "Network simulator (ns-2)," http://www.isi.edu/nsnam/ns/.

[13] S. H. Low, L. Peterson, and L. Wang, "Understanding TCP vegas: a duality model," in *Proceedings of the ACM SIGMETRICS*, 2001, pp. 226–235.

[14] S. H. Low, F. Paganini, and J. C. Doyle, "Ineternet congestion control," in *IEEE Control Systems Magazine*, 2002, pp. 28–43.

[15] E. Altman, T. Jiménez, and R. Núñez Queija, "Analysis of two competing TCP/IP connections," *Perform. Eval.*, vol. 49, pp. 43–55, 2002.

[16] A. Chaintreau, F. Baccelli, and C. Diot, "Impact of TCP-like congestion control on the throughput of multicast groups," *IEEE/ACM Trans. Netw.*, vol. 10, pp. 500–512, August 2002.

[17] M. Garetto, R. L. Cigno, M. Meo, and M. A. Marsan, "Modeling short-lived TCP connections with open multiclass queuing networks," *Computer Networks*, vol. 44, pp. 153–176, February 2004.

[18] S. Liu, T. Başar, and R. Srikant, "Exponential-red: a stabilizing aqm scheme for low- and high-speed TCP protocols," *IEEE/ACM Transactions on Networking*, vol. 13, pp. 1068–1081, October 2005.

[19] J. K. Sundararajan, S. Jakubczak, M. Médard, M. Mitzenmacher, and J. Barros, "Interfacing network coding with TCP: an implementation," Tech. Rep., August 2009, http://arxiv.org/abs/0908.1564.

[20] T. Kelly, "Scalable TCP: improving performance in highspeed wide area networks," *SIGCOMM Comput. Commun. Rev.*, vol. 33, pp. 83–91, April 2003.

[21] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky, "An extension to the selective acknowledgement (sack) option for TCP," United States, 2000.