Modeling Network Coded TCP: Analysis of Throughput and Energy Cost

MinJi Kim*, Thierry Klein†, Emina Soljanin†, João Barros§, Muriel Médard*

***Abstract*—We analyze the performance of TCP and TCP with network coding (TCP/NC) in lossy networks. We build upon the framework introduced by Padhye et al. and characterize the throughput behavior of classical TCP and TCP/NC as a function of erasure probability, 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. It is further seen that TCP/NC has significant throughput gains over TCP.**

In addition, we show that TCP/NC may lead to cost reduction for wireless network providers while maintaining a certain quality of service to their users. We measure the cost in terms of number of base stations, which is highly correlated to the energy, capital, and operational costs of a network provider. We show that increasing the available bandwidth may not necessarily lead to increase in throughput, particularly in lossy networks in which TCP does not perform well. We show that using protocols such as TCP/NC, which are more resilient to erasures, may lead to a throughput commensurate the bandwidth dedicated to each user.

I. 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. 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 [3]–[5]; 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 [5], [6] give an overview and a comparison of various TCP versions over wireless links.

Some relief may come from network coding [7], 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 [8]. There are several practical protocols that take advantage of network coding in wireless networks [9]–[12]. In order to combine the benefits of TCP and network coding, [13] proposes a new protocol called TCP/NC.

In this paper, we present a performance evaluation of TCP as well as TCP/NC in lossy networks. We adopt and extend the TCP model in [2] – i.e. we consider standard TCP with Go-Back-N pipe lining. Thus, the standard TCP discards packets that are out-of-order. We analytically show the throughput gains of TCP/NC over standard TCP. 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. Furthermore, we use NS-2 (Network Simulator [14]) 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.

We use the model for TCP/NC's and TCP's performance to study their effect on cost of operating a network. In particular, we show that maintaining or even improving users' quality of experience may be achieved without installing additional network infrastructure, e.g. base stations. We measure users' quality of experience using the throughput perceived by the user. We make a clear distinction between the terms throughput and *bandwidth*, where throughput is the number of *useful* bits over unit time received by the user and bandwidth is the number of bits transmitted by the base station per unit time. In essence, bandwidth is indicative of the resources provisioned by the service providers; while throughput is indicative of the user's quality of experience. For example, the base station, taking into account the error correction codes, may be transmitting bits at 10 megabits per second (Mbps), i.e. bandwidth is 10 Mbps. However, the user may only receive useful information at 5 Mbps, i.e. throughput is 5 Mbps.

The disparity between throughput and bandwidth used can be reduced by using a transport protocol that is more resilient to losses. One method is to use multiple base stations si-

*M. Kim and M. Médard are with the RLE at the Massachusetts Institute of Technology, MA USA (e-mail: {minjikim, medard}@mit.edu).

†T. Klein and E. Soljanin are with the Alcatel-Lucent Bell Laboratories, NJ USA (e-mail: thierry.klein@alcatel-lucent.com, emina@research.bell-labs.com).

§J. Barros is with the Department of Electrical and Computer Engineering at the University of Porto, Portugal (e-mail: jbarros@fe.up.pt).

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

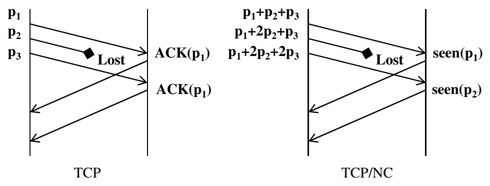


Fig. 1: Example of TCP and TCP/NC. In the case of TCP, the TCP sender receives duplicate ACKs for packet $\mathbf{p_1}$, which may wrongly indicate congestion. However, for TCP/NC, the TCP sender receives ACKs for packets $\mathbf{p_1}$ and $\mathbf{p_2}$; thus, the TCP sender perceives a longer round-trip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

multaneously (using multiple TCP connections [15] or multipath TPC [16]). However, the management of the multiple streams or paths may be difficult, especially in lossy networks. Furthermore, each path or TCP stream may still suffer from performance degradation in lossy environments [15], [16]. We show that erasure-resilient protocols such as TCP/NC [13], [17] can effectively reduce the disparity between throughput and bandwidth.

There has been extensive research on modeling and analyzing TCP's performance [18]–[23]. 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.

This paper is based on the work from [17], [24]. The paper is organized as follows. In Section II, we provide a brief overview of TCP/NC. In Section III, we introduce our communication model. Then, we provide throughput analysis for TCP and TCP/NC in Sections IV and V, respectively. In Section VI, we provide simulation results to verify our analytical results in Sections IV and V. In Section VII, we present our model for analyzing the cost associated with operating the network, and analyze the number of base stations needed in Section VIII. We study the best case scenario in Section IX, and compare this idealized scenario with those of TCP and TCP/NC in Section X. Finally, we conclude in Section XI.

II. Overview of TCP/NC

Reference [13] introduces a new *network coding* layer between 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 [13], [25]. 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.

III. 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 TCP/NC 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 [4] modifies the congestion control mechanism for networks with high bandwidth-delay products. Other variants include those with selective acknowledgment schemes [3]. It may be interesting to compare the performance of the TCP variants with that of TCP/NC. However, we focus on traditional TCP here.

A. 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.

B. 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 TCP/NC close their window; thus, TCP/NC is able to react to congestion.

C. Performance metric

We analyze the performance of TCP and TCP/NC 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 $\mathcal{T}_{nc}$ to be the average throughput for TCP and TCP/NC, respectively.

D. Intuition

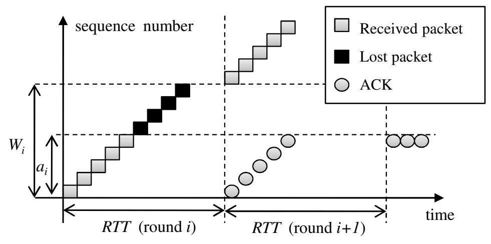
For traditional TCP, random erasures in the network can lead to triple-duplicate ACKs. For example, in Figure 2a, the sender transmits W_i packets in round i; however, only a_i of them arrive at the receiver. As a result, the receiver ACKs the a_i packets and waits for packet $a_i + 1$. When the sender 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 TCP/NC 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.

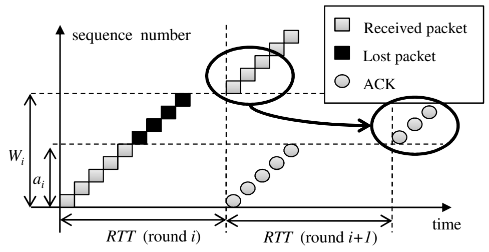
Note that network coding does not mask correlated (or bursty) losses due to congestion. With enough correlated losses, network coding cannot correct for all the losses. As a result, the transmission rate will be adjusted according to standard TCP's congestion control mechanism when TCP/NC detects correlated 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.

IV. Throughput 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



(a) TCP



(b) TCP/NC

Fig. 2: The effect of erasures: TCP experiences triple-duplicate ACKs, and results in $W_{i+2} \leftarrow W_{i+1}/2$. However, 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.

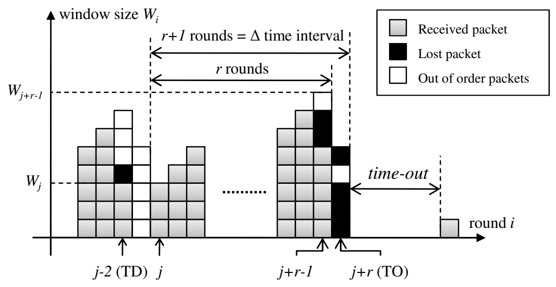


Fig. 3: TCP's window size with a TD event and a TO event. In round $j-2$, losses occur resulting in triple-duplicate ACKs. On the other hand, in round $j+r-1$, losses occur; however, in the following round $j+r$ losses occur such that the TCP sender only receives two-duplicate ACKs. As a result, TCP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

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.

A. 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 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)):

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

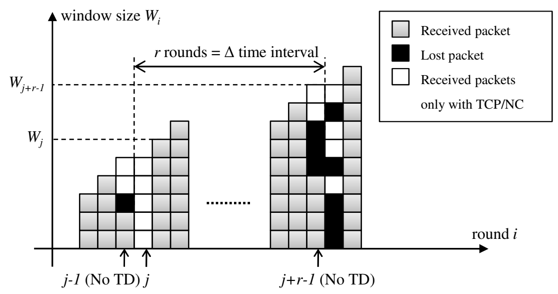


Fig. 4: TCP/NC's window size with erasures that would lead to a triple-duplicate ACKs event when using standard TCP. Note that unlike TCP, the window size is non-decreasing.

The average throughput can be expressed as

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

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:

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

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

B. 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

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

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).

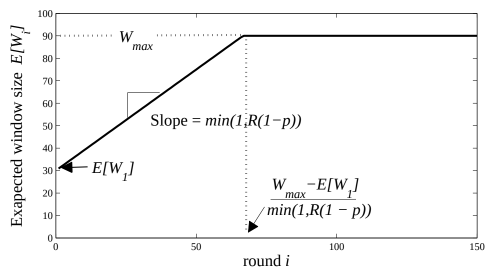


Fig. 5: Expected window size for TCP/NC where $W_{\max} = 90$, $E[W_1] = 30$. We usually assume $E[W_1] = 1$; here we use $E[W_1] = 30$ to exemplify the effect of $E[W_1]$.

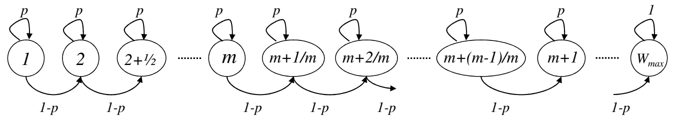


Fig. 6: Markov chain for the TCP/NC's window evolution.

V. Throughput Analysis for TCP/NC

We consider the expected throughput for TCP/NC. Note that erasure patterns that result in TD and/or TO events under TCP may not yield the same result under TCP/NC, as illustrated in Section III-D. 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 TCP/NC to advance its window. This implies that TCP/NC does not experience window closing owing to random losses often.

A. TCP/NC Window Evolution

From Figure 4, we observe that TCP/NC is able to maintain its window size despite experiencing losses. This is partially because TCP/NC is able to receive packets that would be considered out of order by TCP. As a result, TCP/NC'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

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

Fig. 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.

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

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

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

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 TCP/NC's expected window size:

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

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 TCP/NC window using Equation (17).

1) Markov Chain Model: The above analysis describes the expected behavior of TCP/NC'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 TCP/NC 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 III-D, TCP/NC does not often experience a window shutdown, which implies that there are correlated or heavy losses. Thus, TCP/NC's window size is non-decreasing, as shown in Figure 6. Therefore, given enough time, TCP/NC 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:

$$T_p(W) = \sum_{m=S(1)}^{S(W)} N(1,m) = \sum_{m=S(1)}^{S(W)} \frac{1}{1-p} = \frac{1}{1-p}\sum_{m=S(1)}^{S(W)} 1$$
$$= \frac{W(W-1)}{2(1-p)}. \tag{18}$$

$T_p(W)$ 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 V-B1.

B. TCP/NC Analysis per Round

Using the results in Section V-A, 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 TCP/NC. 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, TCP/NC sender transmits $R\cdot E[W_i]$ coded packets, and we expect $pR\cdot E[W_i]$ packets to be lost. Thus, the TCP/NC receiver only receives $(1-p)\cdot R\cdot E[W_i]$ degrees of freedom.

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 [13], [25] 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 V-A1.

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 TCP/NC. 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 TCP/NC is unable to *fully* recover from losses in any given round. By *fully* recover, we mean that TCP/NC decoder is able to acknowledge all packet transmitted in that round. As we shall show in Section VI, TCP/NC 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 $R \geq \frac{1}{1-p}$.

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 TCP/NC experiences.

In lossy networks, TCP/NC'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.

C. TCP/NC Average Throughput

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

$$\mathcal{T}_{nc} = \frac{1}{n}\sum_{i=1}^{n} \frac{E[W_i]}{SRTT}\min\{1, R(1-p)\} = \frac{1}{n \cdot SRTT} \cdot f(n), \quad (20)$$

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 $\mathcal{T}_{nc} \to \frac{W_{\max}}{SRTT}$.

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 TCP/NC to experience a TD or TO event, depending on the variants of TCP used. In Section VI-C, we present simulation results that show that TCP's congestion control mechanism still applies to TCP/NC when appropriate.

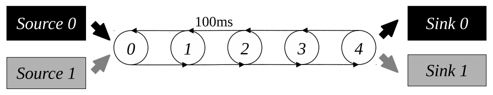


Fig. 8: Network topology for the simulations.

VI. Simulation Results for Throughput Analysis

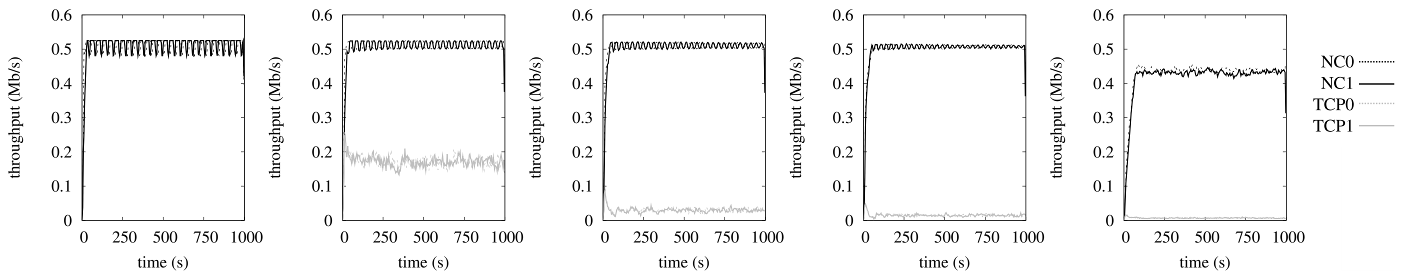
We use simulations to verify that our analysis captures the behavior of both TCP and TCP/NC. We use NS-2 (Network Simulator [14]) to simulate TCP and TCP/NC, where we use the implementation of TCP/NC from [25]. Two FTP applications (ftp0, ftp1) wish to communicate from the source (src0, src1) to sink (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 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 TCP/NC.

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).

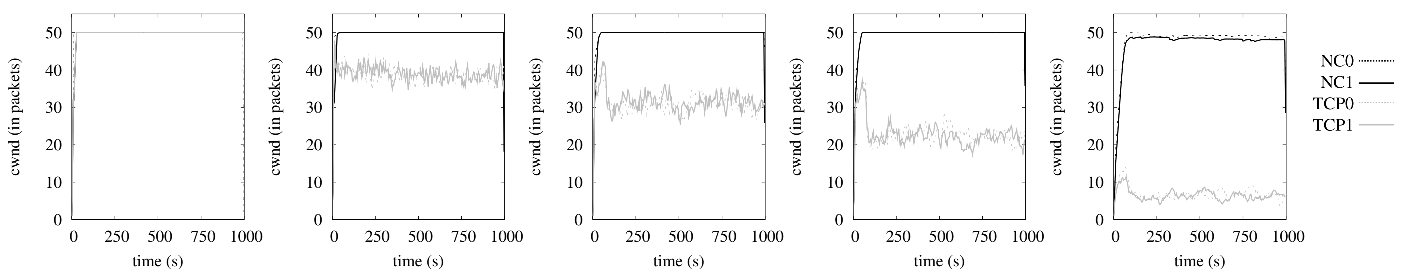
We study the effect these variables have on the following:

- $\mathcal{T}$: Throughput of TCP or TCP/NC,
- $E[W]$: Average window size of TCP or TCP/NC,



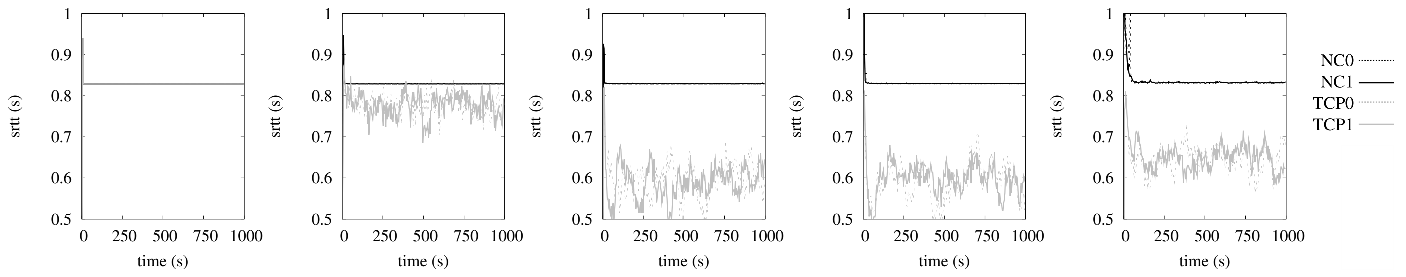
(a) $p = 0$ (b) $p = 0.0199$ (c) $p = 0.0587$ (d) $p = 0.0963$ (e) $p = 0.1855$

Fig. 9: Throughput of TCP/NC and TCP with varying link erasure probability p.



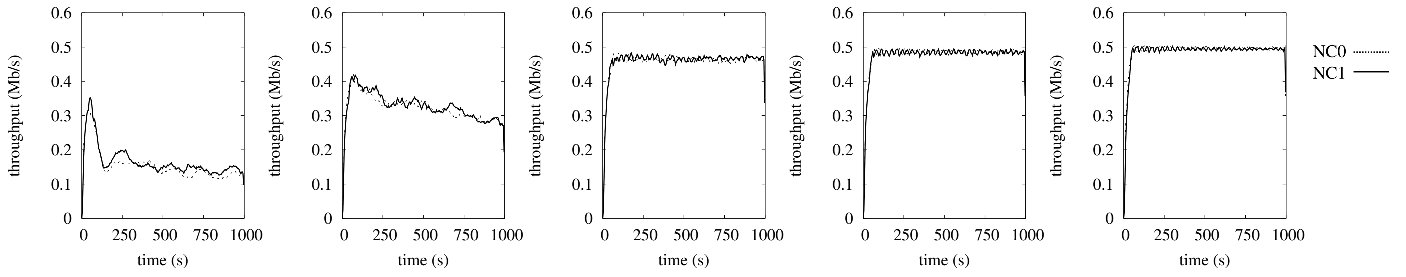
(a) $p = 0$ (b) $p = 0.0199$ (c) $p = 0.0587$ (d) $p = 0.0963$ (e) $p = 0.1855$

Fig. 10: The congestion window size of 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) $p = 0.1855$

Fig. 11: The round trip time estimate (SRTT) of 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) $R = 1.15$

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

- $SRTT$: 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.

A. 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 TCP/NC 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.

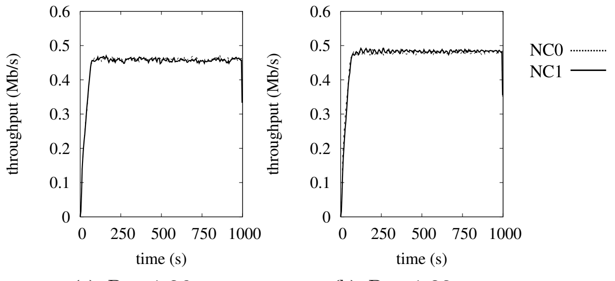
The more interesting result is when $p > 0$. As our analysis predicts, TCP/NC sustains its high throughput despite the random erasures in the network. We observe that TCP may close its window due to triple-duplicates ACKs or timeouts; however, TCP/NC is more resilient to such erasure patterns. Therefore, TCP/NC 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 III-D and V-B2, TCP/NC masks errors by translating losses as longer RTT. For TCP/NC, 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, TCP/NC is able to maintain a consistent estimate of the RTT; however, is slightly above the actual 800 ms.

B. 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 V-B1, 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 TCP/NC to maintain its high throughput. However, we shall show that R need not be much larger than $\frac{1}{1-p}$ for TCP/NC to achieve its maximal throughput.



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

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

In Figure 12, we present TCP/NC 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 $R = 1.11$, TCP/NC achieves a significantly higher throughput than TCP (in Figure 9d) for $p = 0.0963$.

Figure 9e shows that, with $p = 0.1855$, TCP/NC 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 TCP/NC 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.

C. Congestion Control

We showed that TCP/NC 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. 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, TCP/NC is able to use the available bandwidth efficiently. With TCP/NC flows, there is another parameter we need to consider: the redundancy factor R. Since TCP/NC sends R coded packets for each data packet,

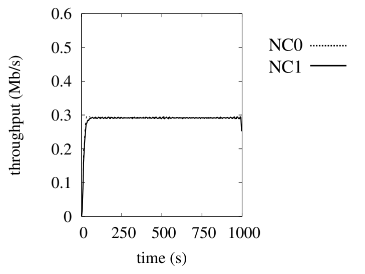


Fig. 14: TCP/NC for $p = 0.0963$ and $C = 0.7$ Mbps.

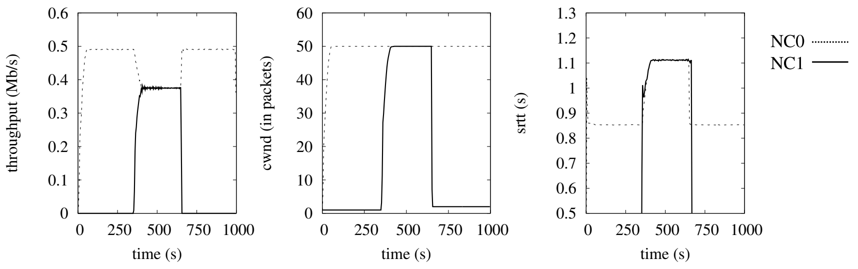


Fig. 15: TCP/NC-TCP/NC for $p = 0.0963$ with congestion ($C = 0.9$ Mbps, $R = 1.2$, $W_{\max} = 50$).

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

p	TCP/NC $SRTT$	R	NC0	NC1	TCP/NC 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

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 TCP/NC can achieve rate close to $C(1-p)$, which is optimal.

We show that multiple TCP/NC 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 $W_{\max}$.

D. Comparison to the analytical model

Finally, we examine the accuracy of our analytical model in predicting the behavior of TCP and TCP/NC. 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 TCP/NC 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 I. We observe that Equations (19) and (17) predict well the trend of TCP/NC's throughput and window evolution, and provides a good estimate of TCP/NC's performance. Furthermore, our analysis predicts the average TCP behavior well. In Table I, 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 TCP/NC is resilient to erasures; thus, better suited for reliable transmission over unreliable networks, such as wireless networks.

VII. Model for Network Cost

Mobile data traffic has been growing at an alarming rate with some estimating that it will increase more than 25-folds in the next five years [26]. In order to meet such growth, there has been an increasing effort to install and upgrade the current networks. As shown in Figure 16, mobile service providers often install more infrastructure (e.g. more base stations) in areas which already have full coverage. The new infrastructure is to provide more bandwidth, which would lead to higher quality of experience to users. However, this increase in bandwidth comes at a significant energy cost as each base station has been shown to use 2-3 kilowatts (kW) [27]. The sustainability and the feasibility of such rapid development have been brought to question as several trends indicate that the technology efficiency improvements may not be able to keep pace with the traffic growth [27].

In the subsequent sections, we use the results from Sections IV, V, and VI to show that TCP/NC allows a better use of the base stations installed and can improve the goodput without any additional base stations. Improving the goodput with the same or a fewer number of base stations implies reduction in energy cost, operational expenses, capital expenses, and maintenance cost for the network provider. The results in this paper can also be understood as being able to serve more users or traffic growth with the same number of base stations. This may lead to significant cost savings, and may be of interest for further investigation.

A. Model

Consider a network with N users. We assume that these N users are in an area such that a single base station can cover them as shown in Figure 16. If the users are far apart enough that a single base station cannot cover the area, then more base stations are necessary; however, we do not consider the problem of coverage.

The network provider's goal is to provide a *fair* service to any user that wishes to start a transaction. Here, by fair, we mean that *every user is expected to be allocated the same average bandwidth*, denoted as $\mathcal{B}$ Mbps. The network provider wishes to have enough network resources, measured in number of base stations, so that any user that wishes to start a transaction is able to join the network immediately and is given an average bandwidth of $\mathcal{B}$ Mbps. We denote $\mathcal{T}$ to be the throughput seen by the user. Note that $\mathcal{T} \leq \mathcal{B}$.

We denote N_{bs} to be the number of base stations needed to meet the network provider's goal. We assume that every base station can support at most $\mathcal{B}_{\max}$ Mbps (in bandwidth) and at most $N_{\max}$ active users simultaneously. In this paper, we assume that $\mathcal{B}_{\max} = 300$ Mbps and $N_{\max} = 200$. As previously, we denote p to be the probability of packet loss in the network, and RTT to be the round-trip time.

A user is *active* if the user is currently downloading a file; *idle* otherwise. A user decides to initiate a transaction with probability q at each time slot. Once a user decides to initiate a transaction, a file size of f bits is chosen randomly according to a probability distribution Q_f. We denote μ_f to be the expected file size, and the expected duration of the transaction to be $\Delta = \mu_f/\mathcal{T}$ seconds. If the user is already active, then the new transaction is added to the user's queue. If the user has initiated k transactions, the model of adding the jobs into the user's queue is equivalent to splitting the throughput $\mathcal{T}$ to k transactions (each transaction achieves a throughput of $\mathcal{T}/k$ Mbps).

VIII. Analysis of the Number of Base Stations

We analyze N_{bs} needed to support N users given bandwidth $\mathcal{B}$ and throughput $\mathcal{T}$. We first analyze $P(\Delta, q)$, the probability that a user is active at any given point in time. Given $P(\Delta, q)$, we compute the expected number of active users at any given point in time and N_{bs} needed to support these active users.

To derive $P(\Delta, q)$, we use the Little's Law. For a stable system, the Little's Law states that the average number of jobs (or transactions in our case) in the user's queue is equal to the product of the arrival rate q and the average transaction time Δ. When $\Delta p \geq 1$, we expect the user's queue to have on average at least one transaction in the long run. This implies that the

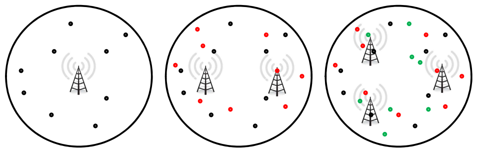


Fig. 16: As number of users in a given area grows, a service provider may add additional base stations not for coverage but for bandwidth. As red users join the network, a second base station may be necessary; as green users join the network, a third base station may become necessary in order to maintain a certain level of quality of service.

user is expected to be active at all times. When $\Delta p < 1$, we can interpret the result from Little's Law to represent the probability that a user is active. For example, if $\Delta p = 0.3$, the user's queue is expected to have 0.3 transactions at any given point in time. This can be understood as the user being active for 0.3 fraction of the time. Note that when the system is unstable, the long term average number of uncompleted jobs in the user's queue may grow unboundedly. In an unstable system, we assume that in the long term, a user is active with probability equal to one.

Therefore, we can state the following result for $P(\Delta, q)$.

$$P(\Delta, q) = \min\{1, \Delta q\} = \min\left\{1, \frac{\mu_f}{\mathcal{T}} \cdot q\right\}. \quad (21)$$

Given $P(\Delta, q)$, the expected number of active users is $NP(\Delta, q)$. We can now characterize the expected number of base stations needed as

$$N_{bs} = NP(\Delta, q) \cdot \max\left\{\frac{\mathcal{B}}{\mathcal{B}_{\max}}, \frac{1}{N_{\max}}\right\}. \quad (22)$$

In Equation (22), $\max\{\frac{\mathcal{B}}{\mathcal{B}_{\max}}, \frac{1}{N_{\max}}\}$ represents the amount of base stations' resources (the maximum load $\mathcal{B}_{\max}$ or the amount of activity $N_{\max}$) each active user consumes. The value of N_{bs} from Equation (22) may be fractional, indicating that actually $\lceil n_{bs} \rceil$ base stations are needed.

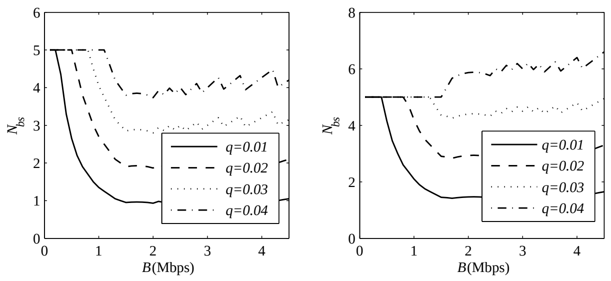
Note the effect of $\mathcal{B}$ and $\mathcal{T}$. As shown in Equation (22), increasing $\mathcal{B}$ incurs higher cost while increasing $\mathcal{T}$ reduces the cost. Therefore, when a network provider dedicates resources to increase $\mathcal{B}$, the goal of the network provider is to increase $\mathcal{T}$ proportional to $\mathcal{B}$.

IX. Best Case Scenario

In an ideal scenario, the user should see a throughput $\mathcal{T} = \mathcal{B}$. In this section, we analyze this best case scnario with $\mathcal{T} = \mathcal{B}$. This assumption can be considered as ignoring the effect of losses in the network; thus, TCP or TCP/NC can achieve the full bandwidth available. Once we understand the optimal scenario, we then consider the behavior of TCP and TCP/NC in Section X.

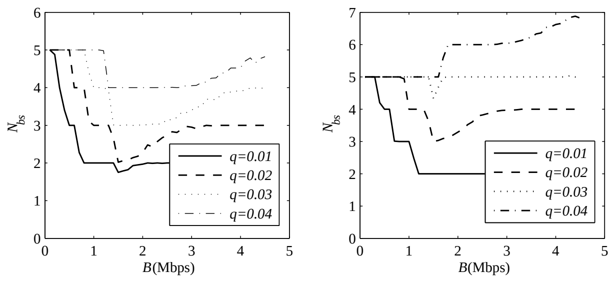
A. Analytical Results

In Figures 17a and 17b, we plot Equation (22) with $\mu_f = 3.2$ MB and $\mu_f = 5.08$ MB for varying values of q. As



(a) μ_f = 3.2 MB (b) μ_f = 5.08 MB

Fig. 17: The values of N_{bs} from Equation (22) with $N = 1000$ and varying q and $\mathcal{B}$.



(a) μ_f = 3.2 MB (b) μ_f = 5.08 MB

Fig. 18: Average value of N_{bs} over 100 iterations with $N = 1000$ and varying q and $\mathcal{B}$.

$\mathcal{B}$ increases, it does not necessarily lead to increase in N_{bs}. Higher $\mathcal{B}$ results in users finishing their transactions faster, which in turn allows the resources dedicated to these users to be released to serve other requests or transactions. As a result, counter-intuitively, we may be able to maintain a higher $\mathcal{B}$ with *the same or a fewer* number of base stations than we would have needed for a lower $\mathcal{B}$. For example, in Figure 17a, when $\mathcal{B} < 1$ Mbps, the rate of new requests exceeds the rate at which the requests are handled; resulting in an unstable system. As a result, most users are active all the time, and the system needs $\frac{n}{N_{\max}} = \frac{1000}{200} = 5$ base stations.

There are many cases where N_{bs} is relatively constant regardless of $\mathcal{B}$. For instance, consider $q = 0.03$ in Figure 17b. The value of N_{bs} is approximately 4-5 throughout. However, there is a significant difference in the way the resources are used. When $\mathcal{B}$ is low, all users have slow connections; therefore, the base stations are fully occupied not in bandwidth but in the number of active users. On the other hand, when $\mathcal{B}$ is high, the base stations are being used at full-capacity in terms of bandwidth. As a result, although the system requires the same number of base stations, users experience better quality of service and users' requests are completed quickly.

When q and $\mathcal{B}$ are high enough, it is necessary to increase N_{bs}. As demand exceeds the network capacity, it becomes necessary to add more infrastructure to meet the growth in demand. For example, consider $q = 0.04$ in Figure 17b. In this case, as $\mathcal{B}$ increases N_{bs} increases.

B. Simulation Results

We present MATLAB simulation results to verify our analysis results in Section IX-A. We assume that at every 0.1 second, a user may start a new transaction with probability $\frac{q}{10}$. This was done to give a finer granularity in the simulations; the results from this setup is equivalent to having users start a new transaction with probability q every second. We assume that there are $N = 1000$ users. For each iteration, we simulate the network for 1000 seconds. Each plot is averaged over 100 iterations.

Once a user decides to start a transaction, a file size is chosen randomly in the following manner. We assume there are four types of files: f_{doc} = 8KB (a document), f_{image} = 1MB (an image), f_{mp3} = 3 MB (a mp3 file), f_{video} = 20 MB (a small video), and are chosen with probability q_{doc}, q_{image}, q_{mp3}, and q_{video}, respectively. In Figure 18a, we set $[q_{doc}, q_{image}, q_{mp3}, q_{video}] = [0.3, 0.3, 0.3, 0.1]$. This results in $\mu_f = 3.2$ MB as in Figure 17a. In Figure 18b, we set $[q_{doc}, q_{image}, q_{mp3}, q_{video}] = [0.26, 0.27, 0.27, 0.2]$, which gives $\mu_f = 5.08$ MB as in Figure 17b.

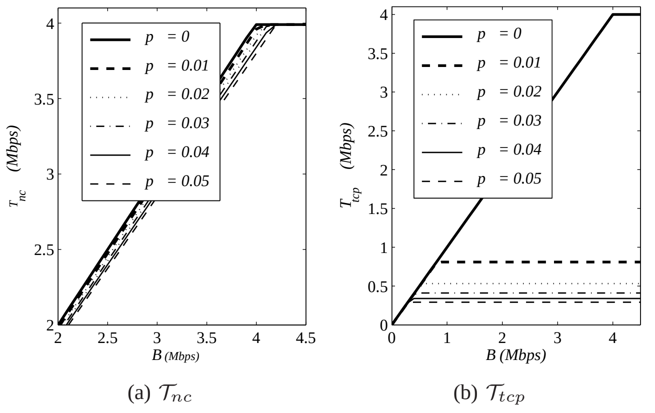
The simulation results show close concordance to our analysis. Note that the values in Figures 18a and 18b are slightly greater than that of Figures 17a and 17b. This is because, in the simulation, we round-up any fractional N_{bs}'s since the number of base stations needs to be integral.

X. Analysis for the Number of Base Stations for TCP/NC and TCP

We now study the effect of TCP and TCP/NC's behavior (i.e. $\mathcal{T} \leq \mathcal{B}$). We have shown that TCP/NC is robust against erasures; thus, allowing it to maintain a high throughput despite random losses. For example, if the network allows for 2 Mbps per user and there is 10% loss rate, then the user should see approximately $2 \cdot (1 - 0.1) = 1.8$ Mbps. Reference [17] has shown, both analytically and with simulations, that TCP/NC indeed is able to achieve throughput close to 1.8 Mbps in such a scenario while TCP fails to do so.

We use the model and analysis from Sections IV, V, and VI. As in Section VI, we set the maximum congestion window, $W_{\max}$, of TCP and TCP/NC to be 50 packets (with each packet being 1000 bytes long), and their initial window size to be 1. We consider RTT = 100 ms and varying p from 0 to 0.05. We note that, given $\mathcal{B}$ and p, $\mathcal{T} \leq \mathcal{B}(1 - p)$ regardless of the protocol used.

Combining Equation (20) and $\mathcal{T}_{nc} \leq \mathcal{B}(1 - p)$, we obtain the values of $\mathcal{T}_{nc}$ for various $\mathcal{B}$, RTT, and p. In Figure 19a, the values of $\mathcal{T}_{nc}$ plateaus once $\mathcal{B}$ exceeds some value. This is caused by $W_{\max}$. Given $W_{\max}$ and RTT, TCP/NC and TCP both have a maximal throughput it can achieve. With the parameters we are considering, the maximal throughput is approximately 4 Mbps. Note that regardless of p, all TCP/NC flows achieve the maximal achievable rate. This shows that TCP/NC can overcome effectively the erasures or errors in the network, and provide a throughput that closely matches



(a) $\mathcal{T}_{nc}$ (b) $\mathcal{T}_{tcp}$

Fig. 19: The value of $\mathcal{T}_{nc}$ and $\mathcal{T}_{tcp}$ against $\mathcal{B}$ for varying values of p. We set RTT = 100 ms.

the bandwidth $\mathcal{B}$.

Combining Equation (16) and $\mathcal{T}_{tcp} \leq \mathcal{B}(1-p)$, we obtain the values of $\mathcal{T}_{tcp}$ for various $\mathcal{B}$, RTT, and p as shown in Figure 19b. As in Figure 19a, the values of $\mathcal{T}_{tcp}$ are also restricted by $W_{\max}$. However, TCP achieves this maximal throughput only when p = 0. This is because, when there are losses in the network, TCP is unable to recover effectively from the erasures and fails to use the bandwidth dedicated to it. For $p > 0$, $\mathcal{T}_{tcp}$ is not limited by $W_{\max}$ but by TCP's performance limitations in lossy wireless networks.

Using the results in Figure 19 and Equation (22), we can obtain the number of base stations N_{bs} needed by both TCP and TCP/NC as shown in Figures 20 and 21. TCP suffers performance degradation as p increases; thus, N_{bs} increases rapidly with p. Note that increasing $\mathcal{B}$ without being able to increase $\mathcal{T}$ leads to inefficient use of the network, and this is clearly shown by the performance of TCP as $\mathcal{B}$ increases with non-zero loss probability, $p > 0$.

However, for TCP/NC, N_{bs} does not increase significantly (if any at all) when p increases. As discussed in Section VIII, TCP/NC is able to translate better $\mathcal{B}$ into $\mathcal{T}_{nc}$ despite $p > 0$, i.e. $\mathcal{B} \approx \mathcal{T}_{nc}$. As a result, this leads to a significant reduction in N_{bs} for TCP/NC compared to TCP. Note that N_{bs} for TCP/NC is approximately equal to the values of N_{bs} in Section VIII regardless of the value of p. Since TCP/NC is resilient to losses, the behavior of $\mathcal{T}_{nc}$ does not change as dramatically against p as that of $\mathcal{T}_{tcp}$ does. As a result, we observe N_{bs} for TCP/NC to reflect closely the values of N_{bs} seen in Section VIII, which is the best case with $\mathcal{B} = \mathcal{T}$.

We observe a similar behavior for other values of RTT as we did for RTT = 100 ms. The key effect of the value of RTT in the maximal achievable throughput. For example, if $W_{\max}$ is limited to 50, the maximal achievable throughput is approximately 0.8 Mbps when RTT = 500 ms, which is much less than the the 4 Mbps achievable with RTT = 100 ms. As a result, for RTT = 500 ms, neither $\mathcal{T}_{nc}$ nor $\mathcal{T}_{tcp}$ can benefit from the increase in $\mathcal{B}$ beyond 0.8 Mbps. Despite this limitation, TCP/NC still performs better than TCP when losses occur. When demand exceeds the maximal achievable throughput, N_{bs} increases for both TCP/NC and TCP in the same manner. We do not present the results for want of space.

XI. Conclusions

We have presented an analytical study and compared the performance of TCP and TCP/NC. Our analysis characterizes the throughput of TCP and TCP/NC as a function of erasure probability, 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. TCP/NC 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 TCP/NC's performance. Our analysis and simulation results both support that TCP/NC is robust against erasures and failures. Thus, TCP/NC is well suited for reliable communication in lossy wireless networks.

In addition, we studied the number of base stations N_{bs} needed to improve the throughput to the users. It may seem that higher throughput necessarily increases N_{bs}. Indeed, if there are enough demand (i.e. high throughput per connection, many active users in the network, etc.), we eventually need to increase N_{bs}. However, we show that this relationship is not necessarily true. When the observed throughput by the user is low, each transaction takes more time to complete and each user stays in the system longer. This degrades the user experience and delays the release of network resources dedicated to the user. This is particularly important as the number of active users each base station can support is limited to the low hundreds. We observed that, given bandwidth allocated a user, achieving low throughput may lead to a significant increase in N_{bs} and an ineffective use of the network resources; while achieving high throughput may lead to reduction in N_{bs}. We showed that TCP/NC, which is more resilient to losses than TCP, may better translate bandwidth to throughput. Therefore, TCP/NC may lead to a better use of the available network resources and reduce the number of base stations N_{bs} needed to support users at a given throughput.

References

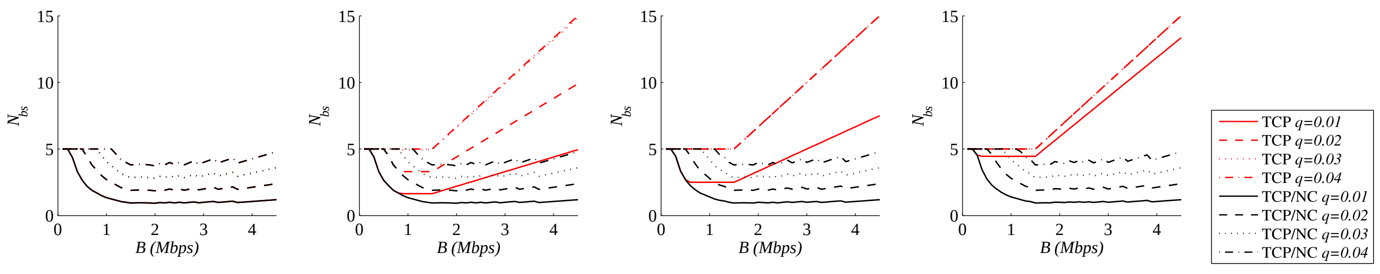
[1] R. Cáceres and L. Iftode, "Improving the performance of reliable transport protocols in mobile computing environments," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 5, June 1995.

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

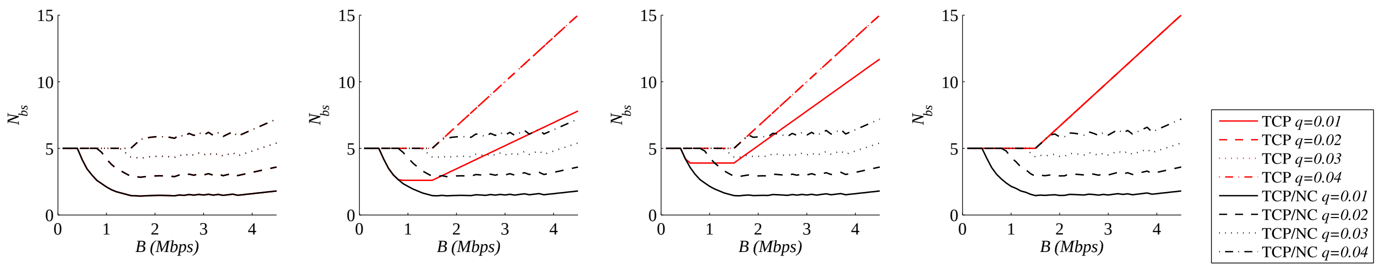
[4] T. Kelly, "Scalable TCP: improving performance in highspeed wide area networks," *SIGCOMM Comput. Commun. Rev.*, vol. 33, pp. 83–91, April 2003.

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



(a) p = 0 (b) p = 0.01 (c) p = 0.02 (d) p = 0.05

Fig. 20: The value of N_{bs} from Equation (22) for TCP and TCP/NC with varying p and q. Here, RTT = 100 ms, $W_{\max}$ = 50, N = 1000, and μ_f = 3.2 MB. In (a), $p = 0$ and both TCP and TCP/NC behaves the same; thus, the curves overlap. Note that this result is the same as that of Figure 17a. In (b), the value of N_{bs} with TCP for $p = 0.03$ and 0.04 coincide (upper most red curve). In (c) and (d), the values of N_{bs} with TCP for $p > 0.01$ overlap.



(a) p = 0 (b) p = 0.01 (c) p = 0.02 (d) p = 0.03

Fig. 21: The value of N_{bs} from Equation (22) for TCP and TCP/NC with varying p and q. Here, RTT = 100 ms, $W_{\max}$ = 50, N = 1000, and μ_f = 5.08 MB. In (a), the results for TCP and TCP/NC are the same. Note that this result is the same as that of Figure 17b. In (b) and (c), the value of N_{bs} with TCP for $p > 0.01$ coincide (upper red curve). In (d), the values of N_{bs} with TCP for any q all overlap. We do not show results for p = 0.04 or 0.05 as they are similar to that of (d).

[6] 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.

[7] 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.

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

[9] 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.

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

[11] Y. Lin, B. Li, and B. Liang, "CodeOr: Opportunisitic routing in wireless mesh networks with segmented network coding," in *Proceedings of IEEE International Conference on Network Protocols*, 2008.

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

[13] J. K. Sundararajan, D. Shah, M. Médard, S. Jakubczak, M. Mitzenmacher, and J. Barros, "Network coding meets tcp: Theory and implementation," *Proceedings of IEEE*, vol. 99, pp. 490–512, March 2011.

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

[15] T. J. Hacker, B. D. Athey, and B. Noble, "The end-to-end performance effects of parallel TCP sockets on a lossy wire-area network," in *Proceedings of the IEEE IPDPS*, 2002.

[16] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural guidelines for multipath TCP development," *IETF, Request for Comments*, no. 6182, March 2011.

[17] M. Kim, M. Médard, and J. Barros, "Modeling network coded TCP throughput: A simple model and its validation," in *Proceedings of ICST/ACM Valuetools*, May 2011.

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

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

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

[21] 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.

[22] 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.

[23] 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.

[24] M. Kim, T. Klein, E. Soljanin, J. Barros, and M. Médard, "Trade-off between cost and goodput in wireless: Replacing transmitters with coding," in *http://arxiv.org/abs/1203.2841*, 2012.

[25] 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.

[26] Cisco, "Cisco visual networking index: Global mobile data traffic forecast," 2011.

[27] D. Kilper, G. Atkinson, S. Korotky, S. Goyal, P. Vetter, D. Suvakovic, and O. Blume, "Power trends in communication networks," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 17, no. 2, pp. 275 –284, 2011.