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 study the cost of improving the goodput per user in a wireless network. We measure the cost in terms of number of base stations, which is highly correlated to the energy cost of a network provider. We show that increasing the available bandwidth may not necessarily lead to increase in goodput, particularly in lossy wireless networks using TCP. We show that using protocols such as TCP/NC, which are more resilient to erasures in the network, may lead to a goodput commensurate with the bandwidth dedicated to each user. By increasing goodput, users’ transactions are completed faster; thus, the resources dedicated to these users can be released to serve other requests, consequently reducing the cost for the network providers.

Keywords Transport protocol · TCP · Wireless · Energy · Network coding

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. Consequently, TCP’s performance in wireless networks is poor when compared to the wired counterparts as shown in [6, 24]. There has been extensive research to combat these harmful effects of erasures and failures [9, 14, 27]; 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 random losses. 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 without receiving acknowledgments.

Some relief may come from network coding [2], 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 erasures [18]. There are several practical protocols that use network coding in wireless networks [5, 7, 13, 19]. In order to combine the benefits of TCP and network coding, [26] proposes a new protocol called TCP/NC.

In this paper, we present a performance evaluation of TCP and TCP/NC in lossy networks. We adopt and extend the TCP model in [24] – i.e. we consider standard TCP with Go-Back-N pipe lining. The standard TCP discards packets that are out-of-order. There are several variants to TCP. STCP [14] modifies the congestion control mechanism for networks with high bandwidth-delay products. Other variants include those with selective acknowledgment [9]. It may be interesting to compare the performance of the TCP variants with that of TCP/NC. However, we focus on traditional TCP here.

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. We also use NS-2 (Network Simulator [1]) to verify our analytical results. Our analysis and simulations show that TCP/NC is robust against erasures. 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. For example, 1-3 % packet loss rate is sufficient to harm TCP’s performance [4, 6, 17, 24]. This performance degradation can lead to inefficient use of network resources and incur substantially higher cost to maintain the same goodput. There has been extensive research to combat these harmful effects of erasures and failures; however, TCP even with modifications does not achieve significant improvement. References [4, 27] give an overview of various TCP versions over wireless links.

The disparity between goodput and bandwidth used can be reduced by using a protocol that is more resilient to losses. One method is to use multiple base stations simultaneously (using multiple TCP connections [12] or multipath TCP [10]). 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 [10, 12]. We show that erasure-resilient protocols such as TCP/NC can effectively reduce the disparity between throughput and bandwidth.

There has been extensive research on modeling and analyzing TCP’s performance [3, 8, 11, 20, 22, 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 [24] 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.

We consider the use of TCP/NC [17, 26] as an alternative transport protocol. TCP/NC may not be the only viable solution, and other transport protocols that can combat erasures may be used. We use TCP/NC for its effectiveness and simplicity. This paper is based on the work from [16, 17].

Overview of TCP/NC: Reference [26] 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 do 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. 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.

2 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 [24]. We denote $W_i$ to be the size of TCP’s congestion control window size at the beginning of round $i$. The sender transmits $W_i$ packets in its window at the start of round $i$, and once all $W_i$ packets have been sent, it defers sending 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 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. Once the sender receives the ACKs, it updates its window size. Assume that $a_i$ packets are acknowledged in round $i$. Then, $W_{i+1} \leftarrow W_i + a_i/W_i$. In general, TCP cannot increase its
window size unboundedly; there is a maximum window size $W_{\text{max}}$. Therefore, $W_{i+1} \leftarrow \max(W_i + a_i/W_i, W_{\text{max}})$.

TCP reduces the window size for congestion control.

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 window, $W_{i+1} \leftarrow 1$. At this point, the sender updates its TO period to $2T_o$ rounds and sends one packet. For any subsequent TO events, the sender transmits the one packet from 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 is zero.

We assume that there 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 [24] 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.

### 2.1 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 $N_{[t_1, t_2]}$ to be the number of packets received by the receiver during the interval $[t_1, t_2]$. The total average throughput is defined as: $\mathcal{T} = \lim_{\Delta \to \infty} \frac{N_{[t_1, t_2]}}{\Delta}$. We denote $T_{\text{tcp}}$ and $T_{\text{nc}}$ to be the average throughput for TCP and TCP/NC, respectively.

### 2.2 Intuition

For traditional TCP, random erasures in the network can lead to triple-duplicate ACKs. For example, in Fig. 1a, 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 Fig. 1b. With network coding, any linearly independent packet delivers new information. Thus, any subsequent packet (in Fig. 1b, 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.

### 3 Throughput analysis for TCP

We consider the effect of losses for TCP. The throughput analysis for TCP is similar to that of [24]. However, the model has been modified from that of [24] 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, in the absence of mechanisms such as selective acknowledgement, often termed SACK. Thus, they are discarded by the TCP receiver. As a result, standard TCP’s throughput behavior with independent losses is similar to that of [24], where losses are correlated within one round.

**Triple-duplicate for TCP:** We consider the expected throughput between consecutive TD events, as shown in
Fig. 1 The effect of erasures: TCP experiences triple-duplicate ACKs, and results in $W_{i+2} = W_{i+1}/2$. However, TCP/NC masks the erasures using network coding, which allows TCP to advance its window.

Fig. 2. 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 $E\left[N_{[t_1, t_2]}\right]$. 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

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

(1)

The packets (white in Fig. 2) sent after the lost packets (black in Fig. 2) are out of order, and will not be accepted by the standard TCP receiver. Thus, Eq. 1 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 Fig. 2, 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, $E[\Delta] = RTT(E[r] + 1)$. To derive $E[r]$, note that $W_{j+r-1} = W_j + r - 1$ and

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

(2)

Equation 2 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 $[0, W_{j-2}]$,

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

(3)

During these $r$ rounds, we expect to successfully transmit $\frac{1-p}{p}$ packets as noted in Eq. 1. This results in: $\frac{1-p}{p} = \left(\sum_{k=0}^{r-2} W_{j+k}\right) + a_{j+r-1} = (r-1)W_j + \frac{(r-1)(r-2)}{2} + a_{j+r-1}$. Taking the expectation and using Eq. 3, $\frac{1-p}{p} = \frac{3}{2}(E[r] - 1)^2 + E[a_{j+r-1}]$. Note that $a_{j+r-1}$ is assumed to be uniformly distributed across $[0, W_{j+r-1}]$. Thus, $E[a_{j+r-1}] = E[W_{j+r-1}]/2 = E[r] - \frac{1}{4}$ by Eq. 3. Solving for $E[r]$, we find:

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

(4)

Fig. 2 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...
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 Eq. 3):

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

(5)

The average throughput can be expressed as

$$T_{tcp} = \frac{E[N_{[i,i+1]}]}{E[\Delta]} = \frac{1 - p}{p} \frac{1}{RTT (E[r] + 1)}.$$  

(6)

For small $p$, $T_{tcp} \approx \frac{1}{RTT} \frac{1}{3p + o(\sqrt{p})}$; for large $p$, $T_{tcp} \approx \frac{1}{RTT} \frac{1}{p}$. If we only consider TD events, the long-term steady state throughput is equal to that in Eq. 6.

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

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

(7)

For small $p$, this result coincide with the results in [24].

**Time-out for TCP:** If there are enough losses within two consecutive rounds, TCP may experience a TO event, as shown in Fig. 2. Thus, $P(\text{TO}|W)$, the probability of a TO event given a window size of $W$, is given by

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

(8)

Note that when the window is small ($W < 3$), then losses result in TO events. For example, assume $W = 2$ with packets $p_1$ and $p_2$ in its window. Assume that $p_2$ is lost. Then, the TCP sender may send another packet $p_3$ in the subsequent round since the acknowledgment for $p_1$ allows it to do so. 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 Eq. 8 with $E[W]$ from Eq. 5. 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 as

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

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

(9)

Finally, by combining the results in Eqs. 7, 8, and 9, we get an expression for the average throughput of TCP as

$$T_{tcp} = \min \left( \frac{W_{\text{max}}}{RTT}, \frac{1 - p}{p} \frac{1}{RTT (E[r] + P(\text{TO}|E[W]) E[\text{duration of TO period}])} \right).$$  

(10)

### 4 Throughput analysis for TCP/NC

The 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 2.2. This is due to the fact that any linearly independent packet conveys a new degree of freedom to the receiver. Figure 3 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.

#### 4.1 TCP/NC window evolution

From Fig. 3, 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 $E[W_i] = E[W_{i-1}] + E[W_{[a_{i-1}]}] = E[W_{i-1}] + \min(1, R(1 - p))$.

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 $E[W_{a_{i-1}}]$ packets to be acknowledged and the window to be incremented by $E[W_{a_{i-1}}]$. Note that $E[W_{a_{i-1}}] = (1 - p) \cdot R \cdot E[W_{i-1}]$ since the encoder transmits on average $R$ linear combinations for every packet transmitted by the TCP sender.

Once we take $W_{\text{max}}$ into account, we have the following expression for TCP/NC’s expected window size:

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

(11)

where $i$ is the round number, $E[W_i]$ is the initial window size, and we set $E[W_1] = 1$.

#### 4.2 TCP/NC analysis per round

Using the results in Section 4.1, we derive an expression for the throughput. The throughput of round $i$, $T_i$, is directly proportional to the window size $E[W_i]$, i.e.
\[ T_i = \frac{E[W_i]}{SRTT} \min\{1, R(1 - p)\} \text{ packets per second,} \quad (12) \]

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 \( T_i \propto (1 - p) \cdot R \cdot E[W_i] \). At any given round \( i \), TCP/NC sender transmits \( R \cdot E[W_i] \) 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.

**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 10\( R \) 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 [25, 26] introduce the redundancy factor with TCP/NC, and show that \( R \geq \frac{1}{1 - p} \) is necessary.

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 5, 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 \( R \geq \frac{1}{1 - p} \).

**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 Eq. 12, 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 \( RTT \). In our model, we assume for simplicity that, despite the losses, \( SRTT \approx RTT \).

### 4.3 TCP/NC average throughput

Taking Eq. 12, we can average the throughput over \( n \) rounds to obtain the average throughput for TCP/NC.

\[
T_{nc} = \frac{1}{n} \sum_{i=1}^{n} E[W_i] \frac{1}{SRTT} \min\{1, R(1 - p)\}
\]

\[
= \frac{1}{n} \sum_{i=1}^{n} \min\left(W_{\text{max}}, E[W_i] + i \min\{1, R(1 - p)\}\right)
\]

\[
= \frac{1}{n} \sum_{i=1}^{n} \min\left(W_{\text{max}}, E[W_i] + i\right) \cdot \frac{n \cdot SRTT}{n \cdot SRTT} \text{ since } R \geq \frac{1}{1 - p}
\]

\[
= \frac{1}{n \cdot SRTT} \cdot f(n),
\]

where

\[
f(n) = \left\{ \begin{array}{ll}
\frac{nE[W_i] + \frac{n(n+1)}{2} \text{ form } \leq r^*}{nW_{\text{max}} - r^*(W_{\text{max}} - E[W_i]) + \frac{r^*(r^*-1)}{2} \text{ form } > r^*}
\end{array} \right.
\]

\[ r^* = W_{\text{max}} - E[W_i]. \]

Note that as \( n \to \infty \), the average throughput \( T_{nc} \to \frac{W_{\text{max}}}{SRTT} \).

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. 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 5, we present simulation results that show that TCP’s congestion control mechanism still applies to TCP/NC when appropriate.

5 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 [1]) 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 Fig. 4. 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 \( q \). There are in total four links in the path from node 0 to node 4, so 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 \( T_o = 3\max\text{RTT} = 3.75 \) rounds long (3 seconds). Therefore, our variables for the simulations are a) the end-to-end erasure rate \( p = 1 - (1 - q)^4 \), b) redundancy factor \( R \), and c) the capacity of the links \( C \) (in Mbps). We study the effect these variables have on 1) throughput of TCP or TCP/NC \( T \), 2) average window size of TCP or TCP/NC \( E[W] \), and 3) round-trip estimate \( SRTT \). For each data point, we average the performance over 100 independent simulation runs, each of which is 1000 seconds long.

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, and 0.015. The corresponding \( p \) values are 0, 0.0199, and 0.0587. The results are shown in Figs. 5, 6, and 7.

Firstly, we show that when there are no random erasures \( (p = 0) \), then TCP/NC and TCP behave similarly, as shown in Figs. 5a, 6a, and 7a. Without any random losses and congestion, all of the flows (NC0, NC1, TCP0, TCP1) achieve the maximal throughput \( \frac{W_{max}}{RTT} \cdot \frac{8}{105} = 0.5 \) 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 Fig. 6).

Interestingly, TCP achieves a moderate average window size although the throughput is much lower (Figs. 5 and 6). This shows that naively 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 2.2 and 4.2, TCP/NC translates 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. The longer RTT estimate takes into account the delay associated with waiting for the next subsequent packet at the receiver. In Fig. 7, we verify that this is indeed true. 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, which is only slightly above the actual 800 ms.

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 4.2, we noted that \( R > \frac{1}{1-p} \) 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 \( R > \frac{1}{1-p} \) 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.

In Fig. 8, we show TCP/NC’s throughput 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 as \( R \) increases from 1.11 to 1.12. Note that \( R = 1.12 \) is only 1% additional redundancy than the theoretical minimum, i.e. \( \frac{1.12}{1/(1-p)} \approx 1.01 \).

A good heuristic in setting \( R \) is the following. Given an erasure probability \( p \) and window size \( W \), the probability that losses in any given round are completely masked is upper bounded by \( \sum_{x=0}^{\frac{W(R-1)}{RTT}} (\binom{R}{x})p^x(1-p)^{R-W-x} \), i.e. there
Fig. 4  Network topology for the simulations

Fig. 5  Throughput of TCP/NC and TCP with varying erasure probability $p$

Fig. 6  The congestion window size of TCP/NC and TCP with varying link erasure probability $p$

Fig. 7  The round trip time estimate (SRTT) of TCP/NC and TCP with varying link erasure probability $p$

Fig. 8  Throughput of TCP/NC for $p = 0.0963$ with varying redundancy factor $R$. Note that $\frac{1}{1 - p} = 1.107$
TCP/NC for $p = 0.0963$ and $C = 0.7$ Mbps

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 in finding an appropriate $R$.

**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 $C$ and $p$, 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. With TCP/NC flows, we need to consider the redundancy factor $R$. Since TCP/NC 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 $R \approx \frac{1}{p}$, 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 $W_{\text{max}} = 50$, $R = 1.2$, $p = 0.0963$, and $C = 0.7$ 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 Fig. 9 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 $R = 1.2 > \frac{1}{1-p} = 1.106$.

### 5.1 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 Eq. 11, demonstrates the same trend as that of the window evolution of TCP/NC NS-2 simulations, shown in Fig. 6. Second, we compare the actual NS-2 simulation performance to the analytical model. This is shown in Table 1. We observe that Eqs. 12 and 11 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 1, we see that Eq. 10 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.

### 6 Model for network cost

In the subsequent sections, we briefly extend the results from Sections 3, 4, and 5, and study how TCP/NC allows a better use of the base stations installed in cellular networks. As shown in Fig. 10, 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) [15]. Improving the

---

**Table 1** The average simulated or predicted long-term throughput of TCP and TCP/NC in megabits per second (Mbps)

| $p$       | TCP/NC $SRTT$ | NC0     | NC1     | TCP0    | TCP1    | TCP analysis |
|-----------|---------------|---------|---------|---------|---------|--------------|
| 0.0963    | 0.0199        | 0.0558  | 0.0297  | 0.0149  | 0.0070  | 0.0098       |
| 0.0964    | 0.0260        | 0.0587  | 0.0325  | 0.0149  | 0.0070  | 0.0098       |
| 0.0965    | 0.0264        | 0.0593  | 0.0329  | 0.0149  | 0.0070  | 0.0098       |
| 0.0966    | 0.0268        | 0.0596  | 0.0332  | 0.0149  | 0.0070  | 0.0098       |
| 0.0967    | 0.0271        | 0.0599  | 0.0335  | 0.0149  | 0.0070  | 0.0098       |
| 0.0968    | 0.0274        | 0.0602  | 0.0338  | 0.0149  | 0.0070  | 0.0098       |
| 0.0969    | 0.0277        | 0.0605  | 0.0341  | 0.0149  | 0.0070  | 0.0098       |
| 0.0970    | 0.0280        | 0.0608  | 0.0344  | 0.0149  | 0.0070  | 0.0098       |
| 0.0971    | 0.0283        | 0.0611  | 0.0347  | 0.0149  | 0.0070  | 0.0098       |
| 0.0972    | 0.0286        | 0.0614  | 0.0349  | 0.0149  | 0.0070  | 0.0098       |

‘NC0’, ‘NC1’, ‘TCP0’, ‘TCP1’ are average throughput achieved in the NS-2 simulations (with the corresponding ‘R’). ‘TCP/NC analysis’ is calculated using Eq. 13 with $\lceil n \cdot SRTT \rceil = 1000$. ‘TCP analysis’ is computed using Eq. 10.
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.

We note that, to prevent TCP’s performance degradation, cellular systems such have implemented various mechanisms (e.g. HARQ [21]) with stringent bit-error rates to reduce packet loss rate. Using a transport protocol that can combat erasures, e.g. TCP/NC, may relieve the lower layers from such stringent performance requirements. It would be interesting to study the effect of using erasure-resilient transport protocols on the lower layers’ performance requirements, and the cross-layer optimization to improve the throughput and the energy cost of cellular systems.

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 Fig. 10. 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 receives the same average throughput, denoted as \( B \) Mbps. It would be interesting to extend and analyze TCP/NC or other alternative protocols under different notions of fairness. However, we use a simple definition of fairness in which all users receive the same throughput. The 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 achieve an average throughput of \( B \) Mbps. We denote \( T \) to be the goodput achieved by the user. Note that \( T \leq B \).

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 \( R_{\text{max}} \) Mbps (in throughput) and at most \( N_{\text{max}} \) active users simultaneously. In this paper, we assume that \( R_{\text{max}} = 300 \) Mbps and \( N_{\text{max}} = 200 \).

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 according to a probability distribution \( Q_f \). We denote \( \mu_f \) to be the expected file size, and the expected duration of the transaction to be \( \Delta = \mu_f / T \) 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 goodput \( T \) to \( k \) transactions (each transaction achieves a rate of \( T/k \) Mbps).

We denote \( p \) to be the probability of packet loss in the network, and \( RTT \) to be the round-trip time. In a wireless, \( p \) and \( RTT \) may vary widely. For example, wireless connection over WiFi may have \( RTT \) ranging from tens of milliseconds to hundreds of milliseconds with loss rates typically ranging from 0-10 %. In a more managed network (such as cellular networks), \( RTT \) are typically higher than that of a WiFi network but lower in loss rates.

### 7 Analysis of the number of base stations

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

Consider a user \( u \) at time \( t \). There are many scenarios in which \( u \) would be active at \( t \). User \( u \) may initiate a transaction at precisely time \( t \) with probability \( q \). Otherwise, \( u \) is still in the middle of a transaction initiated previously.

To derive \( P(\Delta, p) \), 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 transac-
tion time $\Delta$. When $\Delta \rho \geq 1$, we expect the user’s queue to
have on average at least one transaction in the long run. This
implies that the user is expected to be active at all times.
When $\Delta \rho < 1$, we can interpret the result from Little’s Law
to represent the probability that a user is active. For exam-
ple, if $\Delta \rho = 0.3$, the user’s queue is expected to have 0.3
transactions at any given point in time. This can be under-
stood as the user being active for 0.3 fraction of the time.
Note that when the system is unstable, the long term aver-
age 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, $P(\Delta, p) = \min\{1, \Delta \rho\} = \min\{1, \frac{\mu_f}{p}\}$. Then,
the expected number of active users is $n P(\Delta, p)$. We can
now characterize the expected number of base stations
needed as

$$n_{bs} = N \cdot P(\Delta, p) \cdot \max \left\{ \frac{B}{R_{\max}}, \frac{1}{N_{\max}} \right\}$$  \hspace{1cm} (14)

In Eq. 14, $\max \left\{ \frac{B}{R_{\max}}, \frac{1}{N_{\max}} \right\}$ represents the amount of base
stations’ resources (the maximum load $R_{\max}$ or the amount
of activity $N_{\max}$) each active user consumes. The value of
$n_{bs}$ from Eq. 14 may be fractional, indicating that actually
$\lfloor n_{bs}\rfloor$ base stations are needed.

Note the effect of $B$ and $T$. As shown in Eq. 14, increas-
ing $B$ incurs higher cost while increasing $T$ reduces the cost.
Therefore, when a network provider dedicates resources to
increase $B$, the goal of the network provider is to increase $T$
proportional to $B$.

8 Best case scenario, $T = B$

In an ideal scenario, the user should see a goodput $T = B$. In
this section, we analyze this best case scenario with $B = T$.
Once we understand the optimal scenario, we then con-
sider the behavior of TCP and TCP/NC in Section 9 where
generally $T \leq B$.

Analytical Results: In Figs. 11a and b, we plot Eq. 14
with $\mu_f = 3.2$ MB and $\mu_f = 5.08$ MB for varying values
of $q$. As $T$ increases, it does not necessarily lead to increase
in $n_{bs}$. Higher $T$ results in users finishing their transactions
faster, which in turn allows the resources dedicated to these
users to serve other requests or transactions. As a result,
counter-intuitively, we may be able to maintain a higher $T$ with
the same or a fewer number of base stations than we
would have needed for a lower $T$. For example, in Fig. 11a,
when $T < 1$ 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 $\frac{N}{N_{\max}} = \frac{1000}{1000} = 5$ base stations.

There are many cases where $n_{bs}$ is relatively constant
regardless of $T$. For instance, consider $q = 0.03$ in Fig. 11b. The
value of $n_{bs}$ is approximately 4-5 throughout. However,
there is a significant difference in the way the resources are
used. When $T$ is low, all users have slow connections; there-
fore, the base stations are fully occupied not in throughput
but in the number of active users. On the other hand, when
$T$ is high, the base stations are being used at full-capacity
in terms of throughput. 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 $T$ 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 Fig. 11b. In this
case, as $T$ increases $n_{bs}$ increases.

Simulation Results: We present MATLAB simulation
results to verify our analysis results above. We assume
that at every 0.1 second, a user may start a new trans-
action with probability $\frac{q}{10}$. We assume that there are

**Fig. 11** The values of $n_{bs}$ from
Eq. 14 with $N = 1000$ and
varying $q$ and $T$
Fig. 12 Average of \( n_{bs} \) over 100 iterations with \( n = 1000 \), varying \( q \) and \( T \)

\[ \mu_f = 3.2 \text{ MB} \]

\[ \mu_f = 5.08 \text{ MB} \]

\( N = 1000 \) 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 with the mean size \( \mu_f \). In Fig. 12a, \( \mu_f = 3.2 \) MB as in Fig. 11a; in Fig. 12b, \( \mu_f = 5.08 \) MB as in Fig. 11b. The simulation results show close concordance to our analysis. Note that the values in Fig. 12a and b are slightly greater than that of Fig. 11a and b. This is because, in the simulation, we round-up any fractional \( n_{bs} \)'s since the number of base stations needs to be integral.

9 The number of base stations for TCP/NC and TCP

We now study the effect of TCP and TCP/NC’s behavior. We use the model and analysis from Sections 3 and 4 to model the relationship between \( T \) and \( p \) for TCP and TCP/NC, respectively. We set the maximum congestion window, \( W_{\text{max}} \), 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 5 %. We note that, given \( B \) and \( p \), \( T \leq B(1 - p) \) regardless of the protocol used. We use the values of \( T_{nc} \) and \( T_{tcp} \) to compare the number of base stations for TCP/NC and TCP using Eq. 14. We assume that \( SRTT = RTT \). In general, \( SRTT \) is slightly larger than \( RTT \).

Figures 13 and 14 show \( n_{bs} \) predicted by Eq. 14 when \( RTT = 100 \) ms. TCP suffers performance degradation as \( p \) increases; thus, \( n_{bs} \) increases rapidly with \( p \). Note that increasing \( B \) without increasing \( T \) leads to inefficient use of the network, and this is clearly shown by the performance of TCP as \( B \) increases with \( p > 0 \). However, for TCP/NC, \( n_{bs} \) does not increase significantly (if any at all) when \( p \) increases. As discussed in Section 7, TCP/NC is able to translate better \( B \) into \( T_{nc} \) despite \( p > 0 \), i.e. \( B \approx T_{nc} \). As a result, this leads to a significant reduction in \( n_{bs} \) for TCP/NC compared to TCP. Since TCP/NC is resilient to losses, the behavior of \( T_{nc} \) does not change as dramatically against \( p \) as that of \( T_{tcp} \) does. We observe \( n_{bs} \) for TCP/NC to reflect closely the values of \( n_{bs} \) seen in Section 7, which is the best case with \( B = T \).

Fig. 13 The value of \( n_{bs} \) from Eq. 14 for TCP and TCP/NC with varying \( p \) and \( q \). Here, \( RTT = 100 \) ms, \( W_{\text{max}} = 50 \), \( N = 1000 \), and \( \mu_f = 3.2 \) MB.
10 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 goodput $T$ to the users in wireless networks. It may seem that higher $T$ necessarily increases $n_{bs}$. Indeed, if there are enough demand (i.e. $T$, $q$, or $\mu_f$ are high enough), we eventually need to increase $n_{bs}$. However, we show that this relationship is not necessarily true. When $T$ 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.

In wireless networks, the solution to higher demand is often to add more infrastructure. This is indeed necessary if all the base stations are at capacity (in terms of throughput). However, in many cases, the base stations are at capacity either because they are transmitting redundant data to recover from losses; or because they cannot effectively serve more than a few hundred active users. One way to ensure that wireless networks are efficient is to add base stations to effectively increase the goodput of the network. This is costly as base stations are expensive to operate.

References

1. Network simulator (ns-2). http://www.isi.edu/nsnam/ns/
2. Ahlswede R, Cai N, Li SYR, Yeung RW (2000) Network information flow. IEEE Trans Inf Theory 46:1204–1216
3. Altman E, Jiménez T, Núñez Queija R (2002) Analysis of two competing TCP/IP connections. Perform Eval 49:43–55
4. Balakrishnan H, Padmanabhan VN, Seshan S, Katz RH (1997) A comparison of mechanisms for improving TCP performance over wireless links. IEEE/ACM Transactions on Networking 5
5. Barros J, Costa RA, Munaretto D, Widmer J (2009) Effective delay control for online network coding. In: Proceedings of IEEE INFOCOM
6. Cáceres R, Iftode L (1995) Improving the performance of reliable transport protocols in mobile computing environments. IEEE JSAC 13(5):850–857
7. Chachulski S, Jennings M, Katti S, Katabi D (2007) Trading structure for randomness in wireless opportunistic routing. In: Proceedings of ACM SIGCOMM
8. Chaintreau A, Baccelli F, Diot C (2002) Impact of TCP-like congestion control on the throughput of multicast groups. IEEE/ACM Trans Netw 10:500–512
9. Floyd S, Mahdavi J, Mathis M, Podolsky M (2000) An extension to the selective acknowledgement (SACK) option for TCP
10. Ford A, Raiciu C, Handley M, Barre S, Iyengar J (2011) Architectural guidelines for multipath TCP development. IETF, Request for Comments (6182)
11. Garetto M, Cigno RL, Méo M, Marsan MA (2004) Modeling short-lived TCP connections with open multiclass queueing networks. Computer Networks 44:153–176
12. Hacker TJ, Athey BD, Noble B (2002) The end-to-end performance effects of parallel TCP sockets on a lossy wire-area network. In: Proceedings of the IEEE IPDPS
13. Katti S, Rahul H, Hu W, Katabi D, Medard M, Crowcroft J (2006) XORs in the air: Practical wireless network coding. In: Proceedings of ACM SIGCOMM
14. Kelly T (2003) Scalable TCP: improving performance in high-speed wide area networks. SIGCOMM Comput Commun Rev 33:83–91
15. Kilper D, Atkinson G, Korotky S, Goyal S, Vetter P, Suvakovic D, Blume O (2011) Power trends in communication networks. IEEE Journal of Selected Topics in Quantum Electronics 17(2):275–284
16. Kim M, Klein T, Soljanin E, Barros J, Médard M (2012) Trade-off between cost and goodput in wireless: Replacing transmitters with coding. arXiv:1203.2841
17. Kim M, Médard M, Barros J (2011) Modeling network coded TCP throughput: A simple model and its validation. In: Proceedings of ICST/ACM Valuetools
18. Koetter R, Médard M (2003) An algebraic approach to network coding. IEEE/ACM Transaction on Networking 11:782–795
19. Lin Y, Li B, Liang B (2008) CodeOr: Opportunistic routing in wireless mesh networks with segmented network coding. In: Proceedings of IEEE International Conference on Network Protocols
20. Liu S, Başar T, Srikant R (2005) Exponential-RED: a stabilizing AQM scheme for low- and high-speed TCP protocols. IEEE/ACM Trans Net 13:1068–1081
21. Lott C, Milenkovic O, Soljanin E (2007) Hybrid ARQ: Theory, state of the art and future directions. In: Proceedings of IEEE ITW. Bergen, Norway
22. Low SH, Paganini F, Doyle JC (2002) Internet congestion control. IEEE Control Syst Mag:28–43
23. Low SH, Peterson L, Wang L (2001) Understanding TCP Vegas: a duality model. In: Proceedings of the ACM SIGMETRICS, pp 226–235
24. Padhye J, Firoiu V, Towsley D, Kurose J (1998) Modeling TCP throughput: A simple model and its empirical validation. In: Proceedings of the ACM SIGCOMM
25. Sundararajan JK, Jakubczak S, Médard M, Mitzenmacher M, Barros J (2009) Interfacing network coding with TCP: an implementation. Tech rep. arXiv:0908.1564
26. Sundararajan JK, Shah D, Médard M, Jakubczak S, Mitzenmacher M, Barros J (2011) Network coding meets tcp: Theory and implementation. Proceedings of IEEE 99:490–512
27. Tian Y, Xu K, Ansari N (2005) TCP in wireless environments: Problems and solutions. IEEE Comm Mag 43:27–32