Analytic Throughput Model for Network Coded TCP in Wireless Mesh Networks

Sanfeng Zhang[1], Xiang Lan[2] and Shuang Li[2]
[1] Key Laboratory of Computer Network and Information Integration in Southeast University, Ministry of Education
Nanjing, 211189 - China
[2] College of Software Engineering
Southeast University, Nanjing, 211189 - China
[e-mail: sfzhang@seu.edu.cn, xlan@seu.edu.cn, shuangli@seu.edu.cn]
*Corresponding author: Sanfeng Zhang

Received February 13, 2014; accepted August 8, 2014; published September 30, 2014

Abstract

Network coding improves TCP's performance in lossy wireless networks. However, the complex congestion window evolution of network coded TCP (TCP-NC) makes the analysis of end-to-end throughput challenging. This paper analyzes the evolutionary process of TCP-NC against lossy links. An analytic model is established by applying a two-dimensional Markov chain. With maximum window size, end-to-end erasure rate and redundancy parameter as input parameters, the analytic model can reflect window evolution and calculate end-to-end throughput of TCP-NC precisely. The key point of our model is that by the novel definition of the states of Markov chain, both the number of related states and the computation complexity are substantially reduced. Our work helps to understand the factors that affect TCP-NC's performance and lay the foundation of its optimization. Extensive simulations on NS2 show that the analytic model features fairly high accuracy.

Keywords: Wireless mesh networks, transmission control, network coding, markov chain, throughput

A preliminary version of this paper has been accepted by IEEE WCNC 2014, April 6-9, Istanbul, Turkey. This version includes a more comprehensive analysis on related work, a concrete analysis for results and discussions on fairness. This research was supported by a grant of National Natural Science Foundation of China [No. 61300200, Research on Key Technologies for Wireless Mesh Networks Based on Network Coding and Opportunistic Forwarding].

http://dx.doi.org/10.3837/tiis.2014.09.009

1. Introduction

Wireless Mesh Network (WMN) has drawn wide attention in both academia and industry. Despite increased TCP-based WMN services emerge, the TCP protocol itself has poor performance in WMNs [1-2]. Because of the varying conditions and interference, random erasure is much more common in WMNs than that in wired networks. However, TCP protocol mistakenly attributes all the losses to congestion and experiences lots of unnecessary window closing.

One method to address this problem is to modify existing protocols. TCP protocol can be improved by allowing the sender to maintain appropriate window size and proper sending rate when random packet loss occurs. One may also modify the MAC protocol, coordinating each node's channel access opportunity to avoid transmission conflicts as well as to improve spatial utility [3]. These approaches, however, are not practical, since modification of existing protocol stack will cause lots of incompatibility.

Another method is to take advantage of network coding (NC) to relieve TCP from the random link erasures [4-5]. However, these techniques are hard to implement and not TCP-compatible, due to the fact that they don't solve the contradiction between the batch-based coding/decoding operation of NC and the stream nature of TCP transmission.

Sundararajan et al. [6] proposed Network Coded TCP (TCP-NC) and addressed the dilemma tactfully. TCP-NC inserts a coding layer between TCP layer and network layer innovatively. The sender transmits random linear combinations of packets upon receiving an acknowledgement (ACK). ACKs are generated by receivers when new packets are "seen". Every time the receiver receives a linearly independent packet, an ACK will be sent. Therefore, the sender's window can slide properly. Furthermore, Sundararajan et al. introduced the implementation details of TCP-NC based on TCP-Reno in [7].

TCP-NC can be easily implemented based on existing protocols. It handles packet losses through sending redundant coded packets. However, that makes TCP-NC differ from traditional TCP protocols in many aspects such as congestion control, resource scheduling and fairness. Therefore, it is essential to establish a proper model to understand NC's impact on TCP protocol.

This paper models the TCP-NC operational process using a two-dimensional Markov chain. The model establishes the quantitative relation between the end-to-end throughput and TCP-NC parameters, such as erasure rate, coding redundancy, and maximum congestion window size. This work provides theoretical basis for further optimization of TCP-NC.

The rest of the paper is organized as follows: In section 2, we review related work about modeling TCP. In section 3, we provide a brief overview of TCP-NC. In section 4, we discuss our model in detail. In section 5, we verify the accuracy of our model through extensive simulations. Finally, we conclude in Section 6.

2. Related Work

The models of TCP protocol are gradually refined [8-10] in wired networks. With more and more work been done, they becomes increasingly comprehensive and covers a variety of characteristics of different TCP variants. Casetti *et al.* [8] used the Markov chain to describe performance of a single TCP stream, and then expanded it to the cases of multiple TCP streams. Sikdar *et al.* [9] provided a more comprehensive model of Tahoe, Reno and SACK in

wired networks and a more detailed description of timeout phase, slow-start phase and delayed ACK timer was presented, thereby gaining better accuracy. Furthermore, [10] modeled TCP-NewReno, which considered the fast recovery and timeout mechanism. These models, however, only take into consideration the congestion induced losses but not link error induced losses, even though the methods are still of value to be used in wireless networks.

There also has been lots of modeling work about TCP's performance in wireless networks. Abouzeid *et al.* conducted some early research on TCP performance in the presence of random erasures [11]. Li *et al.* considered the impact of 802.11MAC and DSR routing protocol, and used Markov renewal process to model TCP-Reno and TCP- NewReno in wireless networks [12]. Xiao *et al.* proposed a system model for analyzing TCP's performance in multi-hop wireless networks, which took into account the packet buffering, competition for accessing channel and channel's spatial reuse [13]. These work considered packet losses, channel competition and other factors. Based on the proposed models, optimal parameters setup can be obtained. However, these models are only subject to TCP-Reno, TCP-NewReno and other common TCP variants.

When it comes to modeling TCP-NC, there is relatively little work, since NC has further increased the complexity of TCP. Kim *et al.* established a simple model which analyzes the relationship between throughput and maximum window size [14-15]. They simply assumed that redundancy parameter is set large enough to mask all link errors. However, in practice, TCP-NC may coexist with other TCP streams, and various redundancy parameters need to be considered. In these cases, their model becomes unrealistic. By contrast, our model predicts the throughput of TCP-NC under various redundancy parameters and quantifies the evolution of congestion window precisely. Based on it, many mechanisms, such as rate control can be designed to achieve optimization of TCP-NC's fairness and friendliness with other TCP variants.

3. Overview of TCP-NC

In this part, we will analyze TCP-NC's operational process and explain the concepts of accumulative unacknowledged packets and effective window, which are the bases of our model.

We assume that the end-to-end roundtrip time (***RTT***) is far more than the time needed to send all the packets in a window, an assumption also adopted (either implicitly or explicitly) in [11] and [14]. We could thus consider TCP-NC transmits packets in a '*round*' form. The sender transmits all packets in the window at the beginning of each round. Then it stops sending and waits for ACKs. Each time TCP layer puts a packet into the congestion window, the coding layer sends R (redundancy parameter, $R \geq 1$) linear combinations of packets. Each time the receiver receives a linearly independent packet, an ACK will be replied and the sender's congestion window will slide.

Fig. 1 describes the encoding, sending and acknowledging process where coding redundancy parameter is set to 1: when the receiver receives q_1, it will send the ACK of p_1. When it receives q_4, it can get $p_2 = (q_4 - q_1 - p_3 - 2p_4)/3$ so the ACK of p_2 is sent to the sender. We can see that:

- The packets lost in current round can be acknowledged in the next round; ACKs of the current round may be the responses of packets lost in the previous round.
- Timeout will be triggered if all the sent packets are lost in a round.

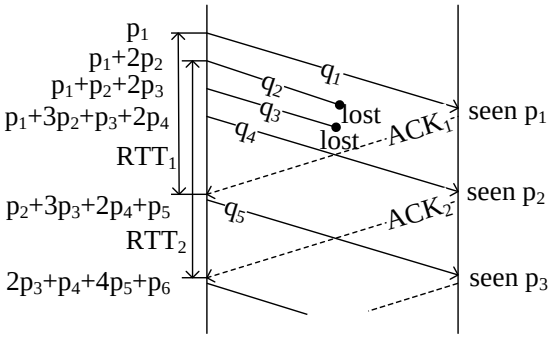


Fig. 1. Coding transmission and acknowledgement in TCP-NC

TCP protocol contains several phases including slow-start, congestion avoidance, fast retransmit, fast recovery and timeout. Congestion Window (***CW***) increases as $CW = CW + 1$ when TCP receives an ACK in the slow-start phase; it shifts into congestion avoidance phase when *CW* is larger than the slow-start threshold (***Ssthresh***) and the *CW* increases as $CW = CW + 1/CW$; it will go back to the slow-start phase with *CW* reset to 1 and *Ssthresh* reset to 2 if some packet is unacknowledged in retransmission-timeout (***RTO***) period. The congestion window size doubles after each round in the slow-start phase and congestion avoidance will start after limited rounds. Therefore, slow-start phase is usually ignored in related modeling work [11, 12 and 14]. Likewise, we focus on TCP-NC's window evolution in congestion avoidance.

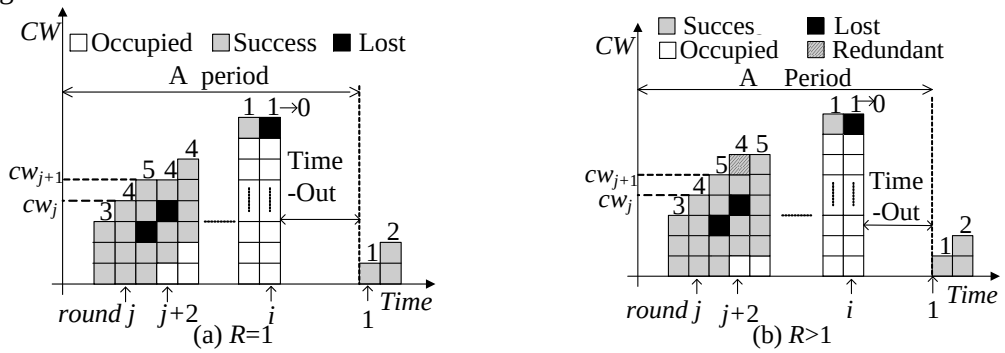


Fig. 2. Evolutionary process of congestion window

We take **Fig. 2** as an example. The horizontal axis is the round of packets and the ordinate axis is congestion window size. TCP-NC's congestion window may evolve in the following way: *CW*=4 in the *j*-th round. If all packets are acknowledged, it will change to 5. We define the number of unacknowledged packets as Accumulative Unacknowledged number (***AU***). If 4 ACKs are returned in (*j*+1)-th round, *CW* will increase to $5\frac{4}{5}$ and *AU* will be 1.Then only 4 new packets can be sent in the (*j*+2)-th round (we define the Effective Window: $\boldsymbol{EW} = \lfloor CW \rfloor - AU$, where $\lfloor x \rfloor$ denotes the maximum integer no greater than *x*). Note that TCP-NC differs from common TCP variants: *CW* may increase, whereas the number of packets can be sent in the window (*EW*) may decrease, since not all packets are ACKed. It seems that only part of the window is effective. In **Fig. 2** (a), a packet is lost in the (*j*+2)-th round, then $CW = 6\frac{2}{6}$ and *AU*=2, thereby *EW* changes to 6-2=4 in the next round. With rounds going on, the effective window is likely to decrease and a timeout will occur if all packets are lost in a round. Then a new period starts with *CW* reset to 1, *EW* to 1 and *AU* to 0.

If $R \geq 1$, AU may not increase. In **Fig. 2** (b), 4 original packets are passed to the congestion window in (j+2)-th round while the coding layer may send 5 linearly independent coded packets. It will successfully transmit 4 coded packets although 1 packet is lost and AU remains to be 1. The congestion window will gradually increase to the maximum if R is set large enough and all these losses are masked.

4. Analytic Throughput model for TCP-NC

4.1 Overview of the model

According to the analysis above, TCP's window evolution is decimal and periodic. However, we only care about two kinds of CW values: 1) CW values increasing to a certain integer for the first time in a period; 2) CW values corresponding to the timeout event of each period. Other CW values are considered as parts of transition. In this way, we can denote the state of TCP-NC as:

$$S = \left\{ (VW, AU) \,\middle|\, \begin{matrix} VW \in N^+, AU \in N^0, AU < VW \\ or \\ VW \in R^+, AU = \lfloor VW \rfloor \end{matrix} \right\} \tag{1}$$

or S(VW, AU) for short, where ***VW*** (Visual Window) is size of the congestion window; AU is the number of unacknowledged packets; the transition between states corresponds to a sequence of packets starts from one state and ends at another.

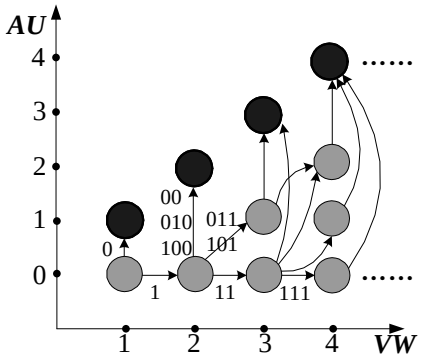


Fig. 3. State transition of Markov chain for TCP-NC

CW may increase to the maximum congestion window size limitation ($CWmax$) when the erasure rate is low. It can still increase visually, but we always set $EW \leq CWmax$. Thus the Visual Window (***VW***) is defined to be distinguished from CW. The process has Markov property because next state is only related to the current one and independent from the previous states. Shown as **Fig. 3**, denoting VW as the horizontal axis, AU as the ordinate axis, we can use a two-dimensional Markov chain to represent the state transition of TCP-NC.

In **Fig. 3**, each black circle stands for a timeout state, or absorbing state. Different absorbing states correspond to different absorbing probabilities. If we denote 0 as a lost packet, 1 as a successfully sent packet, then a sequence of 0/1 starts from one state and ends at another can be obtained. Through analysis of this sequence, we can get the transition probability between any of these states. Some of the sequences between states have been drawn in **Fig. 3**. Transition probability between states is the sum of all probabilities of sequences between them.

Note that the definition of *VW* plays a key role in lowing the model's complexity. It reduces the number of states greatly. Here takes **Fig. 3** as an example. In real evolution, value of *CW* which has $\lfloor CW \rfloor = 3$ can be 3, 3⅓, 3⅔. Thus tuple {*CW*, *AU*} can be {3,0}, {3,1}, {3,2}, {3,3}, {3⅓,0}, {3⅓ ,1}, {3⅓,2}, {3⅓,3}, {3⅔,0}, {3⅔,1}, {3⅔,2}, {3⅔,3}, 12 tuples in total where $\lfloor CW \rfloor = 3$. However, by the state definition as (1), we only need to consider {3,0}, {3,1} and absorbing states {3,3}, {3⅓,3}, {3⅔,3}, therefore more than half of the states are eliminated.

In summary, the following related conclusions hold: 1) a state with $AU = \lfloor VW \rfloor$ is an absorbing state; 2) a state will not transfer to other states with same *VW* value except absorbing states; 3) a state with *VW*=*w* will not transfer to any state with *VW* value larger than *w*+1, it should transfer to some state with *VM*=*w*+1 first; 4) all transitions pointing at none-timeout states end with a successfully sent packet, while all transitions pointing at timeout states end with a lost packet; 5) state *S*(*w*,*w*-1) (*w*≥2) is not included in (1) and **Fig. 3**. Its previous states can only be *S*(*w*-1, *i*) (0≤*i*<*w*-1), and if *AU* increases to *w*-1 earlier than *VW* increases to *w*, *S*(*w*-1, *i*) will transfer to absorbing state *S*(*w*-1, *w*-1), otherwise *S*(*w*-1, *i*) will transfers to *S*(*w*, *j*) (*i*≤*j*≤*w*-2)and will not transfer to *S*(*w*, *w*-1) due to conclusion-2.

According to the total number of packets sent and the time needed, the corresponding throughput can be obtained. The overall throughput of TCP-NC is can be calculated as:

$$E[Thr] = \sum_{w=1}^{M} p_w \cdot \frac{E[rcv_w]}{E[T_w]} \tag{2}$$

where $p_w$means the total probability of absorbing states where $\lfloor VW \rfloor = w$; $E[rcv_w]$ represents the expected number of all the packets sent before timeout; $E[T_w]$ is the expected time of this period, which includes the time of window evolution and timeout. We can imagine that the window could be very large and the states could be too many to calculate. However, many of these states have little probability and are negligible to *E*[*Thr*]. Therefore, we set the upper bound *M* to control the number of states. *M* is often set large enough so that almost all absorbing states are considered. Note that *M* is just an accuracy-related number and has no direct relationship with CW_{max}.

4.2 The absorbing probability p_w and the expected *CW* size

With the absorbing probability p_w , we can calculate the expected *CW* size (*E*[*cwnd*]) an index of sending rate on the sender.

Define $P(\{VM_i, AU_i\} \rightarrow \{VM_j, AU_j\})$ as transition probability from $S(VM_i, AU_i)$ to $S(VM_j, AU_j)$, and $P(S(VM, AU))$ as the transition probability from initial state to *S*(*VM*, *AU*).There are too many paths to calculate p_w, but we can use Lemma-1 below to transform the calculation into calculating the probability that *VM* can increase to *w* and *w*+1.

Lemma-1: The probability of timeout with $\lfloor VW \rfloor = w$ equals the difference between that of *VW* can increase to w and *w*+1.

Proof:

S(*w*,*w*) can only be transformed from *S*(*w*,*i*) (0≤*i*<*w*), so it's the sum of all these transition probability. Thus, we have:

$$\begin{aligned} p_w &= \sum_{i=0}^{w-2} P(S(w,i)) \cdot P(\{w,i\} \rightarrow \{w,w\}) \\ &= \sum_{i=0}^{w-2} P(S(w,i)) \cdot (1 - \sum_{j=i}^{w-1} P(\{w,i\} \rightarrow \{w+1, j\})) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=0}^{w-2} P(S(w,i)) - \sum_{i=0}^{w-2} P(S(w,i)) \cdot \sum_{j=i}^{w-1} P(\{w,i\} \to \{w+1, j\}) \\
&= \sum_{i=0}^{w-2} P(S(w,i)) - \sum_{i=0}^{w-2} \sum_{j=i}^{w-1} P(S(w,i)) \cdot P(\{w,i\} \to \{w+1, j\}) \\
&= \sum_{i=0}^{w-2} P(S(w,i)) - \sum_{i=0}^{w-1} P(S(w+1,i))
\end{aligned}
\tag{3}
$$

where $\sum_{i=0}^{w-2} P(S(w,i))$ and $\sum_{i=0}^{w-1} P(S(w+1,i))$ denote the probability that *VM* could increase to *w* and *w*+1 respectively.

Using Lemma-1, we just need to calculate $P(S(w, i))$ to obtain p_w. After sending *w*-1 packets and losing *i*-*j* packets, $S(w\text{-}1, j)$ will evolve into $S(w, i)$, so:

$$
P(S(w,i)) = \sum_{j=0}^{i} P(S(w-1, j)) \cdot C_{w-2+i-j}^{w-2} \cdot p^{i-j} \cdot (1-p)^{w-1} \tag{4}
$$

where $C_{w-2+i-j}^{w-2} \cdot p^{i-j} \cdot (1-p)^{w-1}$ is the probability of sending *w*-1 packets, losing *i*-*j* packets and finally successfully sending a packet to make *VM* increase from *w*-1 to *w*. Note that *w*-2 is gained by the preceding conclusion-5. The algorithm of calculating $P(S(w, i))$ is showed as Algorithm-1 (**Table 1**) and then p_w can be obtained by equation (3).

Table 1. Algorithm-1

Input: CW_{max}**, end-to-end erasure rate** p **and** M **that limits** VM**.**
Output: $P(S(w, i))$ $(1<w\leq M, 0<i\leq w\text{-}2)$

```
Initialize the Markov chain according to CW_max and M
Initialize P(S(w, i)) (1<w≤ M,0<i≤ w-2)
w←2 , i←0, j←0
while w ≤ M do
    while i ≤ w-2 do
        if i ≥ CW_max   //AU greater than CW_max leads to timeout
            P(S(w, i))=0 //AU cannot be greater than CW_max.
        else
            while j ≤ i do
                P(S(w, i))+= P(S(w-1, j))·C_{w-2+i-j}^{w-2} · p^{i-j} ·(1-p)^{w-1}
                j++
            endwhile
        endif
        j←0
        i++
    endwhile
    i←0
    w++
endwhile
```

The expected *CW* size equals the weighted average of *CW*:

$$E[cwnd] = \sum_{w=1}^{M} w \cdot p_w \tag{5}$$

where M is maximum *VM* value set to lower the complexity. Given that *VM* may increase to CW_{max}, and in that case we take *VM* as CW_{max}. A more comprehensive equation is:

$$E[cwnd] = \sum_{w=1}^{M} \min\{CW_{\max}, w\} \cdot p_w \tag{6}$$

4.3 Expected time T_w from $S(1, 0)$ to absorbing state $S(w, w)$

T_w includes $T_{S(1,0)\to S(w,\cdot)}$, which is the transition time from initial state to any state with $VM = w$ (and *AU* may be 0, 1,2, …, w); $\Delta_{t,w}$, which stands for the time needed for a state with $VM=w$ to evolve into the absorbing state and to w, the RTO at window size. Then:

$$E[T_w] = E[T_{S(1,0)\to S(w,\cdot)}] + E[\Delta_{t,w}] + E[t_{o,w}] \tag{7}$$

Compared to $T_{S(1,0)\to S(w,\cdot)}$, $\Delta_{t,w}$ is rather small, so we set $E[\Delta_{t,w},]$ as $(w\text{-}1)/2 \cdot RTT$ for simplicity, which is the median value of all possible values. $t_{w,o}$ varies with *RTT* based on *RTO* estimation mechanism. Here we use $w \cdot RTT$ to approximate $t_{w,o}$.

Intuitively, $E[T_w]$ equals the weighted average time of all the evolutionary paths. Since there are too many paths, its calculation can be very complicated. To lower the complexity, we first calculate the time *VM* increases by one, and then accumulate all these time.

Lemma 2: Define $T_{S(i,\cdot)\to S(i+1,\cdot)}$ as the duration of packets sequence from any state with $VM = i$ and ends at any state with $VM = i+1$. The expected time of evolution from initial state to a $VW=w$ state can be represented as:

$$E[T_{S(1,0)\to S(w,\cdot)}] = \sum_{i=1}^{w-1} E[T_{S(i,\cdot)\to S(i+1,\cdot)}] \tag{8}$$

Proof:

Assume there are m evolutionary paths from $VM = k$ to $VM = k+1$, each with the time T_S and probability $P(T_s)$ (s = 1,2, ..., m); n evolutionary paths from $VM = k+1$ to $VM = k+2$, each with time T_t' and probability $P(T_t')$ (t = 1,2, ..., n) respectively. The total number of paths from $VM = k$ to $VM = k+2$ is $m \times n$, and the expected time from $VM = k$ to $VM = k+2$ is:

$$\begin{aligned} E[T_{S(k,\cdot)\to S(k+2,\cdot)}] &= \sum_{s=1}^{m}\sum_{t=1}^{n} (T_s + T_t^{'}) \cdot P(T_s) \cdot P(T_t^{'}) \\ &= \sum_{s=1}^{m}\sum_{t=1}^{n} T_s \cdot P(T_s) \cdot P(T_t^{'}) + \sum_{s=1}^{m}\sum_{t=1}^{n} T_t^{'} \cdot P(T_t^{'}) \cdot P(T_s) \\ &= \sum_{s=1}^{m} T_s P(T_s) \cdot \sum_{t=1}^{n} P(T_t^{'}) + \sum_{t=1}^{n} T_t^{'} P(T_t^{'}) \cdot \sum_{s=1}^{m} P(T_s) \end{aligned}$$

Since every path arrives at $VM=k+2$, so:

$$\sum_{s=1}^{m} P(T_s) = 1 \ , \quad \sum_{t=1}^{n} P(T_t^{'}) = 1$$

Thus:

$$E[T_{S(k,\cdot)\to S(k+2,\cdot)}] = \sum_{s=1}^{m} T_s P(T_s) + \sum_{t=1}^{n} T_t^{'} P(T_t^{'}) = E[T_{S(k,\cdot)\to S(k+1,\cdot)}] + E[T_{S(k+1,\cdot)\to S(k+2,\cdot)}]$$

Namely, the expected time of evolution from $VM = k$ to $VM = k+2$ is equal to the sum of that from $VM = k$ to $VM = k+1$ and from $VM = k+1$ to $VM = k+2$. Due to k's arbitrariness and using mathematical induction, it's not difficult to prove: the expected time of evolution between a state with $VM=i$ to a state with $VM=j$ equals the sum of each step's expected time in which VM advances by 1. If we let i be 1 and j be w, we can get:

$$E[T_{S(1,0)\to S(w,\cdot)}] = \sum_{i=1}^{w-1} E[T_{S(i,\cdot)\to S(i+1,\cdot)}]$$

The next task is to calculate $E[T_{S(i,\cdot)\to S(i+1,\cdot)}]$. We can get the rounds needed for $VM = i$ to increase to $VW = i+1$:

$$Round_{i\to i+1} = \frac{i}{E[EW_i]} \tag{9}$$

where $E[EW_i]$ is expected effective window size when $VM=w$. Obviously, $E[EW_1]=1$. And if $VW=i$ ($i>1$) and $AU=j$ ($0\le j<i$), $EW=i-j$. Thus:

$$E[EW_i] = \sum_{j=0}^{i-2} (i-j)\cdot P(S(i,j)) \tag{10}$$

If packet loss rate is low, EW_i may exceed CW_{max}. In that case, we take EW_i the same value of CW_{max}:

$$E[EW_i] = \sum_{j=0}^{i-2} \min\{CW_{\max}, (i-j)\}\cdot P(S(i,j)) \tag{11}$$

Substituting (11) into (9), and assuming RTT is the same for each round [9, 11], we get:

$$E[T_{S(i,\cdot)\to S(i+1,\cdot)}] = \frac{i}{\sum_{j=0}^{i-2} \min\{CW_{\max}, (i-j)\}\cdot P(S(i,j))}\cdot RTT \tag{12}$$

And the expression of $E[T_w]$ is:

$$E[T_w] = \left(\sum_{i=2}^{w-1} \frac{i}{\sum_{j=0}^{i-2} \min\{CW_{\max}, (i-j)\}\cdot P(S(i,j))} + \frac{3w+1}{2} \right)\cdot RTT \tag{13}$$

4.4 The expected number of all the packets sent before timeout

In congestion avoidance, after all packets within the window are successfully sent, the window advances by one.

Thus the following holds:

$$rcv_w = \sum_{i=1}^{w-1} i + \Delta_{rcv,w} \tag{14}$$

where $\Delta_{rcv,w}$ is the number of packets sent in the last round. Compared to $\sum_{i=1}^{w-1} i$, it is rather small, so we set its expectation half the size of congestion window for simplicity. Hence:

$$E[rcv_w] = E[\sum_{i=1}^{w-1} i] + E[\Delta_{rcv,w}] = \frac{w^2}{2} \tag{15}$$

Finally, we get the expected throughput of TCP-NC by substituting $E[rcw_w]$, p_w, $E[T_w]$ into (2):

$$E[Thr] = \sum_{w=1}^{M} \left(\frac{w^2/2}{\left(\sum_{i=0}^{w-2} P(S(w,i)) - \sum_{i=0}^{w-1} P(S(w+1,i)) \right) \cdot RTT} \cdot \frac{1}{\sum_{i=2}^{w-1} \left(i / \sum_{j=0}^{i-2} \min\{CW_{\max}, (i-j)\} \cdot P(S(i,j)) \right) + (3w+1)/2} \right) \tag{16}$$

4.5 Influences of redundancy parameter *R*

Loss rate p can be transformed to $1\text{-}R\cdot(1\text{-}p)$ with R=1 in case of R>1. Then we can use the algorithm mentioned above. Note that if p'<0, R may be large enough to compensate for the random erasures and we could use the model in [14], which is suitable in that case.

4.6 Computing accuracy and complexity analysis

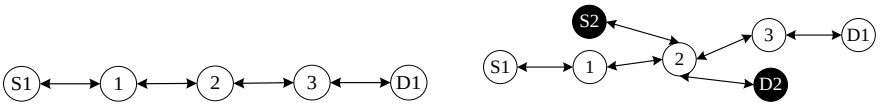
Since *VM* can be very large, Algorithm-1 may be very complex. Given the presence of erasure rate, the probability *VM* being large is rather low and can be negligible. Therefore, the upper bound M is set to help lower the complexity. Algorithm-1 uses a three-ordered nested loop and both the time complexity and space complexity are $O(M^3)$. Note that M is often set about twice of CW_{max}, thus all the computation can be done in nearly constant time.

5 Simulation Results and Model Verification

We verify our model through experiments on NS2 (Network Simulator 2 [16]). After that, a comparison between our model, the models in [11] and [14] will be given to show the accuracy of our model. TCP-NC in our simulation is implemented based on TCP-Reno.

5.1 Network topology and configuration

We conduct our simulations in a four-hop tandem network and a cross-shaped network as in **Fig. 4**. A FTP flow is deployed between the corresponding source and destination. Each link's bandwidth is set to 1Mbps. The transport layer protocol is either TCP-Reno or TCP-NC. Each link's propagation delay is 100ms. Thus for topology (a), link layer *RTT* is 800ms, and for topology (b) the *RTT* between S_1 and D_1 is 800ms while it is 400ms between S_2 and D_2. Each link has the same erasure rate and the packet size is 1KB. Additionally if there is no extra statement, we set all CW_{max} to 50 pkts in the following experiments.



(a) four-hop tandem network (b)cross-shaped network

Fig. 4. Experiment topology

5.2 Model verification under different *R*

When $R > 1$, we use $1\text{-}R\cdot(1\text{-}p)$ to represent the visual erasure rate in R=1 case, and then Algorithm-1 can be used. **Fig. 5** shows that our model and experiments show very close concordance under different values of R.

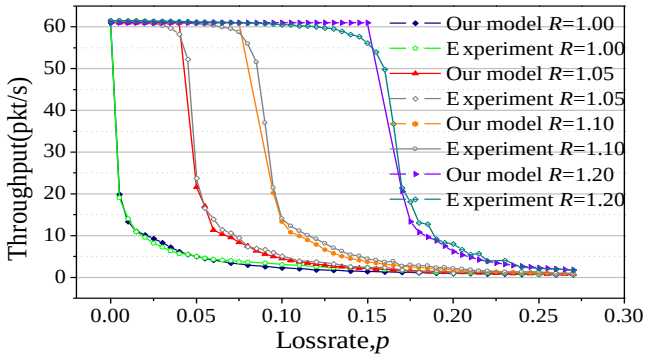


Fig. 5. Influence of redundancy parameters

5.3 Tradeoff between complexity and accuracy

Because the analytic model calculates the weighted mean value of throughput for all visual windows, more visual windows bring higher accuracy and higher computation complexity. The relationship between accuracy and upper bound M is shown in **Fig. 6**. M stands for computation complexity and accumulative absorbing probability P measures the throughput accuracy. It is easy to see that for each erasure rate, P is near to 1 when M exceeds some threshold. It implies that almost all possible visual windows are considered.

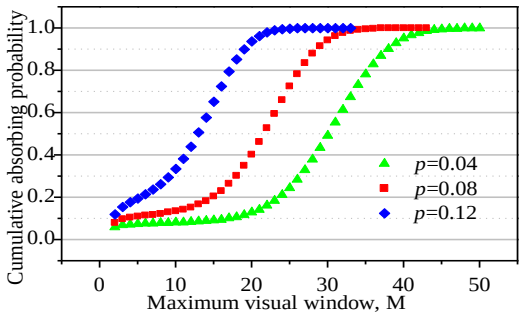


Fig. 6. Complexity and accuracy

5.4 TCP-NC analytic model evaluation

Fig. 7 evaluates our analytic model with simulation results. **Fig. 7** (a, c, e) are outputs of topology (a) and **Fig. 7** (b, d, f) are those of topology (b). **Fig. 7** (a, b) compare average congestion window sizes under different end-to-end erasure rates, and **Fig. 7** (c, d) compare the throughputs under the same condition. In **Fig. 7** (e, f), we set different maximum window sizes and compare throughputs. Note that under the same end-to-end loss rate, both of the flows in **Fig. 7** (b) must have the same average congestion window, and thus the curves must overlap. For that reason, we only draw NC1's curve to verify our model.

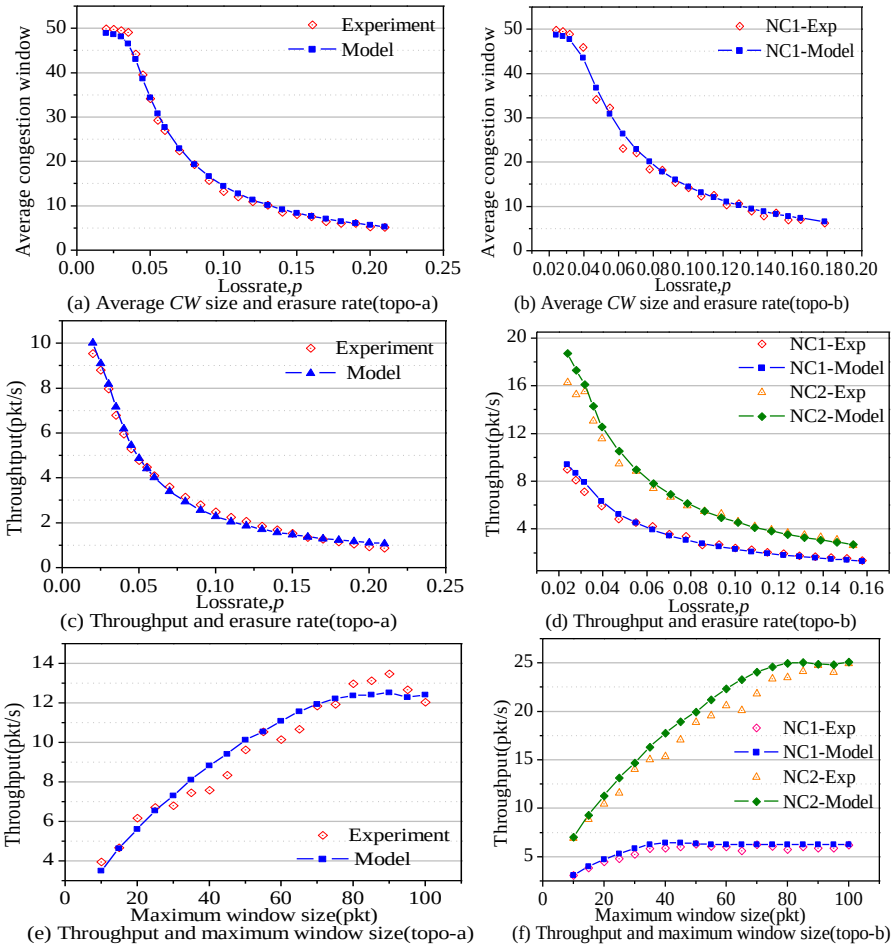


Fig. 7. Comparison between model and experiment results

We can see that our model and experiments match well over a wide range of loss rate (from 0 to 20%) and maximum window size (from 10 to 100 pkts).

5.5 Comparing with other models

We select two similar models for comparison. One is proposed by Abouzeid *et al.* [11], which depict the behavior of TCP-Reno under random erasures. Another is established by Kim *et al.* [14], under the assumption that all random erasures are masked by setting R large enough. Since they both could only calculate the throughput, we compare our model with theirs in

terms of throughput using topology(a). In **Fig. 8**, we present the results of all the models under different erasure rates and redundancy parameters. Kim's model only cares about the maximum window size, and in our experiments, all the maximum window sizes are set the same value (50 pkts) so it has the same curves. The erasure rates in Abouzeid's model are transformed using $p \doteq 1\text{-}R\cdot(1\text{-}p)$ so that both TCP-NC and TCP-Reno have the same visible erasure rates, thus more convincing results can be obtained. As can be seen, Abouzeid's model uses transformed erasure rate to calculate the throughput of TCP-Reno. The throughput is then used to predict TCP-NC's so the deviation may be very large. This also shows that network coding has a significant impact on the behavior of TCP. Kim *et al.*'s model is actually a horizontal line because of its assumption that all random losses are masked. It matches the experiments well when the loss rate is low but deviates farther as the loss rate increases. By contrast, our model precisely predicts the throughput of TCP-NC, not only in a large range of loss rate, but also under different redundancy parameters.

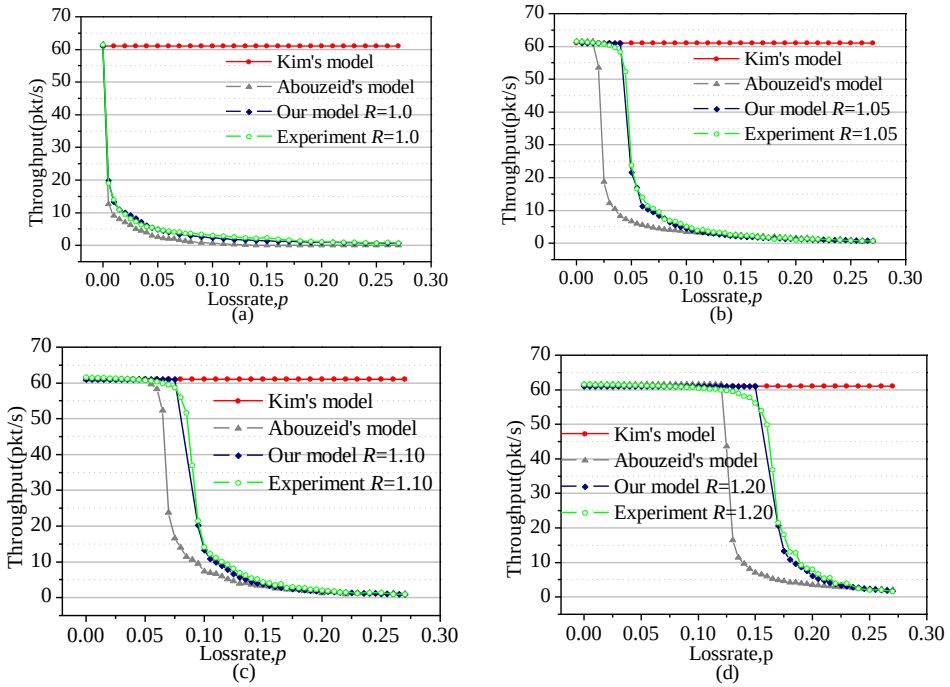


Fig. 8. Accuracy comparison with previous models

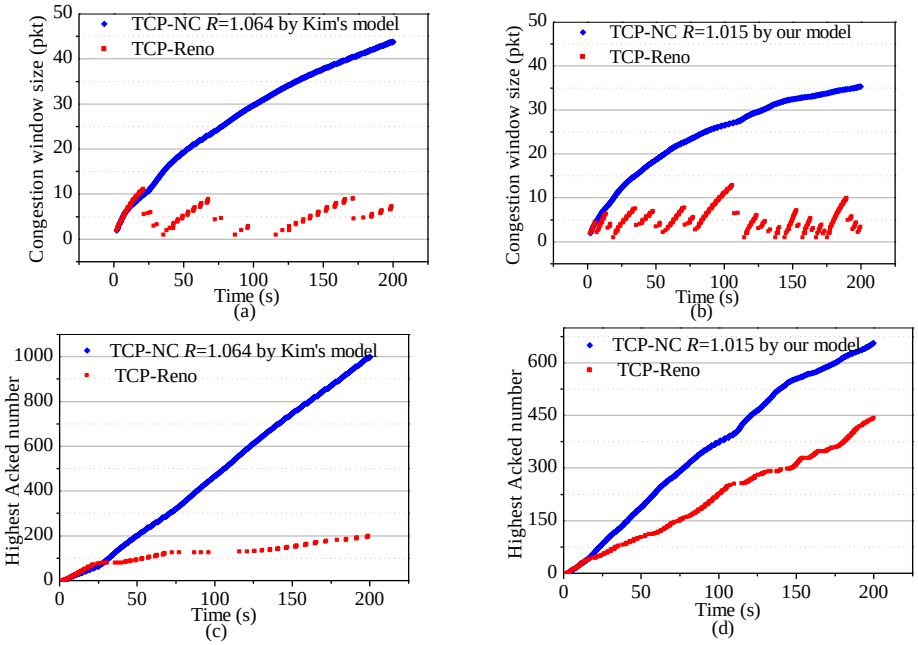


Fig. 9. Performance comparison when TCP-NC coexists with TCP-Reno

5.6 Friendliness optimization based on the analytic model

We can get a better R when TCP-NC coexists with other TCP variants based on our model. We conduct our experiment in the topology in **Fig. 4** (a) to illustrate this point. Without modifying other parameters, we set the bandwidth to be 0.05Mbps and end-to-end erasure rate to be 0.06. The bandwidth is not sufficient if a TCP-NC stream and a TCP-Reno stream coexist. Through the calculation of our model, R should be set to 1.015 thus TCP-NC occupies approximate half the bandwidth. If we simply set R to be $1/(1-p) \approx 1.064$ (as supposed in Kim *et al.*'s model [14]), there is much throughput penalty on TCP-Reno. **Fig. 9** shows R's influence on TCP-NC's bandwidth occupation. **Fig. 9** (a) and (b) are the window evolution process with R set to be 1.064 and 1.015 separately. We emphasize again here that only part of TCP-NC's window is effective because part of packets sent in a round can be Acked. Larger R makes the window of TCP-NC increase faster, but it also makes TCP-Reno experience more window closing. **Fig. 9** (c) and (d) show the evolution of the sequence number of Acked packets. In the case of R=1.064, the sequence number of TCP-Reno rises slowly and intermittently. However, if we set R=1.015, both the Acked sequence number of TCP-NC and TCP-Reno increases stably. The throughput ratio between TCP-NC and TCP-Reno is 4.963 when R=1.064, whereas it is 1.415 when R=1.015. We can see that fairness between coded stream and non-coded one can be improved based on our model.

6 Conclusions

This paper analyzes TCP-NC's performance mathematically. Some interesting features of TCP-NC different from common TCP variants are discussed. The analytic model proposed in this paper precisely calculates TCP-NC throughput and congestion window size under a wide range of end-to-end erasure rate, redundancy parameter and maximum window size. It helps

understand the behavior of TCP-NC and the impact of network coding on TCP. One interesting topic in the future is to examine the behavior when multiple TCP-NC streams coexist.

References

[1] R. C´aceres and L. Iftode, "TCP performance in IEEE-802.11-Based ad hoc networks with multiple wireless lossy links," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 5, pp.1329-1342, June, 2007. Article (CrossRef Link)

[2] Ammar Mohammed Al-Jubari, Mohamed Othman, Borhanuddin Mohd Ali and Nor Asilah Wati Abdul Hamid, "TCP performance in multi-hop wireless ad hoc networks: challenges and solution," *EURASIP Journal on Wireless Communications and Networking* vol. 2011, no. 1, pp. 1-25, December, 2011. Article (CrossRef Link)

[3] Zhenghua Fu, Haiyun Luo, Petros Zerfos, Songwu Lu, Lixia Zhang and Mario Gerla, "The impact of multihop wireless channel on TCP performance," *IEEE Transactions on Mobile Computing*, vol. 4, no. 2, pp. 209-212, March/Aril, 2005. Article (CrossRef Link)

[4] Y. Huang, M. Ghaderi, D. Towsley and W. Gong, "TCP performance in coded wireless mesh networks," in *Proc. of Fifth Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pp. 179-187, Jun 16-20, 2008. Article (CrossRef Link)

[5] Steluta Gheorghiu, Alberto Lopez and Toledo Pablo Rodriguez, "Multipath TCP with network coding for wireless mesh networks," in *Proc. of 2010 IEEE International Conference on Communications (ICC)*, pp. 1-5, May 23-27, 2010. Article (CrossRef Link)

[6] Jay Kumar Sundararajan, Devavrat Shah and Muriel Medrad, "Network coding meets TCP," in *Proc. of 28th IEEE International Conference on Computer Communications* (*INFOCOM*), pp. 280-288, April 19-25, 2009. Article (CrossRef Link)

[7] J. K. Sundararajan, D. Shah, M. M´edard,M. Mitzenmacher, and J. Barros, "Network coding meets TCP theory and implementation," in *Proc. of 30th IEEE International Conference on Computer Communications* (*INFOCOM*), pp. 280–288, April 10-15, 2011. Article (CrossRef Link)

[8] C. Casetti and M. Meo, "A new approach to model the stationary behavior of TCP connections," in *Proc. of 19th IEEE International Conference on Computer Communications* (*INFOCOM*), pp.367 -375, March 26-30 2000. Article (CrossRef Link)

[9] Biplab Sikdar, Shivkumar Kalyanaraman, and Kenneth S. Vastola, "Analytic models for the latency and steady-state throughput of TCP Tahoe, Reno, and SACK," *IEEE/ACM Transactions on Networking*, vol. 11, no. 6, pp. 959-971, December 2003. Article (CrossRef Link)

[10] N. Parvez, A. Mahanti, and C. Williamson, "An analytic throughput model for TCP NewReno," *IEEE/ACM Transactions on Networking*, vol. 18, no. 2, pp. 448-461, April 2010. Article (CrossRef Link)

[11] A. A. Abouzeid, S. Roy and M. Azizoglu, "Stochastic modeling of TCP over lossy links," in *Proc. of 19th IEEE International Conference on Computer Communications* (*INFOCOM*), pp. 1724-1733, March 26-33, 2000. Article (CrossRef Link)

[12] Li X., P. Y. Kong, and K. C. Chua, "TCP performance in IEEE 802.11-based ad-hoc networks with multiple wireless lossy links," *IEEE Transactions on Mobile Computing*, vol. 6, no. 12, pp. 1329-1342, December, 2007. Article (CrossRef Link)

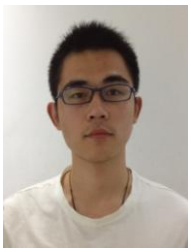
[13] Hannan Xiao, Ying Zhang, James Malcolm, Bruce Christianson and Kee Chaing Chua, "Modeling and analysis of TCP performance in wireless multihop networks," *Wireless Sensor Network*, vol. 2, no. 7, pp. 493-503, July, 2010. Article (CrossRef Link)

[14] M. Kim, M. Médard, and J. Barros, "Modeling network coded TCP throughput: a simple model and its validation," in *Proc. of 4th IEEE International Conference on Software Testing, Verification and Validation*, pp. 131-140, May 16-20, 2011. Article (CrossRef Link)

[15] M. Kim, T. Klein, E. Soljanin, Jo Barros and M. M´edard, "Modeling network coded TCP: analysis of throughput and energy cost," *Technical Report*, *MIT*, 2012.

[16] NS2. http://www.isi.edu/nsnam/ns/, 2013.

Sanfeng Zhang received his Ph.D. degree in Computer Application Technology, from School of Computer Science & Engineering, Southeast University, Nanjing, China, in 2008. Since 2008, Dr. Zhang has been with the College of Software Engineering and School of Computer Science & Engineering, Southeast University, where he is currently an associate professor. His current research interests include network coding and wireless networks.
Email:sfzhang@seu.edu.cn.



Xiang Lan is currently pursuing for the Bachelor's degree in Software Engineering from College of Software Engineering, Southeast University. His current research interests include wireless mesh networks, software defined networks and network performance optimization.
Email: xlan@seu.edu.cn.

Shuang Li received her Bachelor's degree in Software Engineering, from School of Computer Science & Engineering, Southeast University, Nanjing, China, in 2013. She is currently pursuing for the Ph.D. degree in Computer Science and Technology from Tsinghua University. Her current research interests include transparent computing, network modelling and performance optimization.
Email: seu43208301@126.com.