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

MinJi Kim*, Muriel Médard*, João Barros‡

*Massachusetts Institute of Technology
‡University of Porto
TCP in Wireless

• TCP: the dominant protocol in today’s network
• TCP’s congestion control
  • Designed for wireline
  • Mistakes erasures/fading in wireless as congestion, leading to performance degradation in wireless setting
• Use “network coding” to improve performance
TCP using network coding

- Coding layer buffers packets given by TCP
- For every packet coming from TCP, coding layer transmits $R (>1)$ random linear combinations of buffered packets to IP

\[
\begin{align*}
\text{Decoded} & : \\
\begin{array}{cccccccc}
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\end{align*}
\]

- Acknowledgment: ACK a packet upon seeing it (even before it is decoded)
- Can do this at intermediate nodes as well in a daisy chain (tandem) network
- Optional feature: Re-encoding at intermediate nodes.
  - We shall focus on no re-encoding today; end-to-end coding. [Sundararajan et al. ‘09 ‘10]
TCP using Network Coding

\[ RLC = \alpha_1 p_1 + \alpha_2 p_2 + \alpha_3 p_3 + \alpha_4 p_4 + \alpha_5 p_5 \]

Base = first un-ACKed byte

\[ p_1 \]
TCP SubHeader | Data

\[ p_k \]
TCP SubHeader | Data

\[ p_n \]
TCP SubHeader | Data

2 2 4 1 4 2 1 2 2 1
Source Port | Dest Port | Base | \( n \) | Start_1 | End_1 | \( \alpha_1 \) | Start_2 | End_2 | \( \alpha_2 \) | ... 

Obtained from TCP header

Coding coefficient 

\( n \) times
Network Coding with TCP (TCP/NC)

- Random linear network coding masks link losses from TCP in order to prevent unnecessary back-off.
- Translates the losses as longer round trip time.
- ACK design that accounts for mixing (coding) of packets with each other.
  - ACK: every innovative linear combination, even if it does not reveal a packet immediately.
- For every packet coming from TCP, coding layer transmits $R (>1)$ random linear combinations of buffered packets to IP.
Main Idea of TCP/NC

- TCP mistakes losses as congestion; Window closing due to:
  - Triple-duplicate ACKs.
  - Time-out.

- TCP/NC substitutes the lost packets with subsequent packets; avoiding window closing.

Fig. 2: The effect of erasures: TCP experiences triple-duplicate ACKs, and results in $W_{i+2} \leftarrow W_{i+1}/2$. However, E2E-TCP/NC masks the erasures using network coding, which allows TCP to advance its window. This figure depicts the sender’s perspective, therefore, it indicates the time at which the sender transmits the packet or receives the ACK.
Example: No losses

- Increment window by 1
- Sliding window

When no losses, network coding doesn’t provide benefits (erasure correction)
Example: Random losses (Triple-Duplicate ACKs)

TCP

\[ p_1 p_2 p_3 p_4 \]
\[ p_2 p_3 p_4 p_5 \]
\[ p_2 p_3 \]

Window closing: \( W \leftarrow W/2 \)

\[ \text{ACK}(p_1) \]
\[ \text{ACK}(p_1) \]

Triplet-duplicate ACKs!

TCP/NC

\[ p_1 p_2 p_3 p_4 \]
\[ \sum p \]
\[ \sum p \]
\[ \sum p \]
\[ \sum p \]

\[ p_4 p_5 p_6 p_7 \]
\[ \sum p \]
\[ \sum p \]
\[ \sum p \]
\[ \sum p \]

\[ p_7 p_8 p_9 p_{10} p_{11} \]
\[ p_8 p_9 p_{10} p_{11} p_{12} \]

- Can’t increment window by 1
- Partial sliding window

Prevents random losses being interpreted as congestion!

There is a lag in the “SEEN” acks: To avoid lag, introduce redundancy!
Example: Congestion/Correlated losses (Time-outs)

TCP

sender window

\( p_1 p_2 p_3 p_4 \)

Waiting…

\( p_2 p_3 p_4 p_5 \)

Time-out!

Window closing:

\( W \leftarrow 1 \)

ACK(\( p_1 \))

TCP/NC

sender window

\( p_1 p_2 p_3 p_4 \)

Waiting…

\( p_2 p_3 p_4 p_5 \)

Time-out!

Window closing:

\( W \leftarrow 1 \)

Still allows congestion control while masking random losses!
Analysis and Simulations

- Based on Padhye et al.’s model (correlated losses to model congestion).

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

\[
\mathcal{T}_{e2e} = \frac{1}{n} \sum_{i=1}^{n} \frac{E[W_i]}{SRTT} \min\{1, R(1-p)\}
\]

\[
= \sum_{i=1}^{n} \frac{\min(W_{max}, E[W_1] + i)}{n \cdot SRTT}
\]

Degrades super-linearly

If redundancy \( R \) appropriately chosen, scales with window size.

\( R \geq 1/(1-p) \) theoretically optimal, which we verify experimentally as well.
Analysis for TCP

TD: triple-duplicate ACKs
TO: timeout

Each round consists of:
- Sender sends packets in its cwnd
- Sender waits for ACKs
- Sender receives at least one ACK for the packets sent

Consider two “loss (TD or TO)” events: e.g. round $j$ to round $j+r$

Expected # of successfully sent packets in those $r$ rounds (given $p$) is $\frac{1-p}{p}$

Expected value of $r$ is $\frac{2}{3} + \sqrt{-\frac{1}{18} + \frac{2}{3} \frac{1-p}{p}}$

By taking into account the time-out period and exponential back-offs:

$$T_{tcp} = \min \left( \frac{W_{\max}}{RTT}, \left( \frac{1-p}{p} \right) \frac{1}{RTT \left( \frac{5}{3} + \sqrt{-\frac{1}{18} + \frac{2}{3} \frac{1-p}{p}} + P(TO|E[W])E[duration \ of \ TO \ period] \right)} \right)$$
Network coding allows out of order packets to be delivered as consecutive degrees of freedom → effectively allowing Selective ACKs (SACK)

- Window progresses despite “loss” events
- Every round, expected window growth is \( \min\{1, R(1 - p)\} \)
- The expected throughput is average of per round throughput (expected window/RTT)

\[
T_{e2e} = \frac{1}{n} \sum_{i=1}^{n} \frac{E[W_i]}{SRTT} \min\{1, R(1 - p)\}
= \frac{\sum_{i=1}^{n} \min(W_{\max}, E[W_1] + i)}{n \cdot SRTT} \quad \text{since} \quad R \geq \frac{1}{1 - p}
\]
Simulations using NS-2

- Max achievable throughput (per flow): 0.5 Mbps
- Each plot/data point is averaged over 100 simulation runs, each 1000 seconds long
- Erasures only at last hop, i.e. wireless
- End-to-end encoding
  - There can be further throughput and energy benefits by using re-encoding

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

assume lossless links
TCP/NC is able to grow its throughput and maintain high rate despite losses.

TCP’s window size is larger compared to its actual throughput – TCP sender is waiting for ACKs.
Redundancy Factor $R$

- $p = 0.0963 \rightarrow 1/(1-p) \approx 1.107$
- Throughput increases dramatically, and the connection stabilizes when $R \geq 1.12$ (which is only 1% more than $1/(1-p)$)
Congestion Control with TCP/NC

- \( p = 0.0963, \, R = 1.2 \)
- Capacity = 0.9 Mbps
- Two connections NC0 and NC1 share the connection fairly (each 0.37 Mbps)
- TCP/NC allows congestion control while masks erasure!
Conclusions

• For TCP
  – random losses (e.g. wireless) have similar effect as correlated/bursty losses, since TCP treats any losses as congestion.

• For TCP/NC
  – throughput decreases proportional to loss rate;
  – when there are enough correlated losses, TCP/NC would time-out and reduce rate to avoid congestion.

• Therefore, TCP/NC is able to maintain high throughput in lossy networks; making it suitable for reliable communication in wireless networks.