ANALYSIS OF ECN ON TCP PERFORMANCE ENHANCING PROXY PERFORMANCE FOR SATELLITE NETWORKS

Ming-Jye Sheng, Kun I. Park, and Richard Manfredary, The MITRE Corporation
Thomas Mak, US Army, PM WIN-T

ABSTRACT

TCP has been a robust and efficient protocol for data communication in the terrestrial wired networks for more than a couple of decades now. However, TCP is very inefficient over high delay and high-error-rate links that might be found in tactical networks especially tactical SATCOM networks. The Space Communications Protocol Standards (SCPS) organization has proposed a more efficient SCPS-TP protocol, which is a modification of TCP to be used for space environments. The Selective Negative Acknowledgement (SNACK) is an important capability provided by SCPS-TP. Because TCP/IP has gone through many incremental refinements, the protocol itself has grown increasingly complex, which makes analytical modeling quite difficult. As a result, much of the evaluation of TCP/IP QoS mechanism has been done using event driven simulators, such as NS-2 or OPNET. More recently, effort for analytical modeling of the performance of TCP has been active. This analysis has been triggered by the limitations inherent in event driven simulations:

- Time consuming for network-centric system engineer to use a packet level granularity when investigating the performance of large, complex networks.
- Analytical models help network engineer understand the effectiveness of new mechanisms such as TCP performance enhancing proxy and/or ECN being added to the system and implement better requirements and test cases coverage for different network deployment scenarios.

SCPS-TP is typically available from Performance Enhancement Proxy (PEP) which serves as a transport layer proxy to split TCP connections. Explicit Congestion Notification (ECN) is used by routers to signal congestion to end points. The analytical model provides quantitative results to compare the ability of ECN signals vs PEP to distinguish between the packet losses due to network congestion and those due to interference or blockages frequently encountered in tactical wireless networks.

1. INTRODUCTION

The Transport Control Protocol (TCP), which was defined in the late 70s and first standardized in 1981, is by far one of the most robust transport protocols that we use in today’s Internet. TCP was defined keeping in view mainly the terrestrial or ground networks. It has been significantly modified over time as more improved versions (Tahoe, Reno, and new Reno) are added. Features of these TCP versions are summarized in Table 1.

<table>
<thead>
<tr>
<th>Versions</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early TCP</td>
<td>Go-back-n model</td>
</tr>
<tr>
<td>Tahoe</td>
<td>Slow start, congestion avoidance, fast retransmit</td>
</tr>
<tr>
<td>Reno</td>
<td>Fast recovery</td>
</tr>
<tr>
<td>New Reno</td>
<td>Eliminate the wait for a retransmit timer when multiple packets are dropped from one window at the expense of retransmitting at most one dropped packet per roundtrip time.</td>
</tr>
</tbody>
</table>

The major extensions of TCP for Satellite communication in Space Communication Protocol Standard Transport Part (SCPS-TP) [1,2] include data transfer methods according to the functional specification defined in RFC 793 with modifications as mentioned in RFC 1122, timestamps and window scaling options adopted from RFC 1323, and the congestion control algorithms from TCP Vegas. SCPS-TP is typically available from Performance Enhancement Proxy (PEP) which serves as a transport layer proxy to split TCP connections.
Several options of SCPS-TP help improve the link utilization and throughput performance in high bit error environments. Among them, Selective Negative Acknowledgement (SNACK) is the option that will be modeled in this paper. During communication, the receiver of data informs the sender of the segments that it did not receive. This option can include information about more than one segment, which makes it invaluable in a long-delay network. The Sender, on receiving a SNACK, aggressively retransmits all the segments that indicate holes in the receiver queue. These aggressive retransmissions prevent retransmission time-outs, which cost more in terms of the link idle time compared to unnecessary retransmissions in terms of wasted bandwidth.

Explicit Congestion Notification (ECN) is used by routers to signal congestion to end points. This paper is an effort to understand PEP and ECN improvement for satellite networks from an analytical view. The rest of the paper is organized as follows. Section 2 details the analytical model of congestion control for TCP-NewReno protocol, and also discusses how it is extended to model the TCP-SNACK option. Furthermore, the model of TCP-SNACK interaction with other QoS mechanism, such as ECN, is also included in Section 2. Section 3 provides analytical results derived from these models. The concluding remarks and discussion are given in Section 4.

2. ANALYTICAL MODELS

A wide variety of analytical techniques have been applied to the problem of TCP modeling with a fair amount of success. These techniques summarized in [3] range over fluid models, Markov chains, and renewal theory. Our models follow the framework proposed by Casetti and Meo [4, 6] who take the novel approach of separating the modeling of network behavior from the modeling of the behavior within a TCP source and then allowing the two to interact via feedback.

We start by modeling congestion control function of NewReno and then modify it with the SNACK option in SCPS-TP. The reason for implementing SNACK by extending the TCP-NewReno, though the conformance requirements for SCPS-TP say Tahoe (RFC 1122), is because most of the current SCPS-TP protocol implementations use slow start and congestion avoidance with the fast retransmit and fast recovery, as mentioned in NewReno. The SCPS-TP requirements were written down long before NewReno was introduced.

The most basic state variables, congestion window and slow start threshold are used to describe slow start and congestion avoidance:

- $W$ – is the congestion window and limits the amount of data that can be sent by the sender.
- $W_i$ - The slow start threshold is used to determine whether the slow start or congestion avoidance algorithm must be used to control data transmission.

The window size distribution is modeled using the Markov process $\{U_k\}$ where $k$ is the epoch of the change in window size. The process is comprised of four types of states, Normal (N), Idle (I), Timeout (T) and Fast Retransmit (F). The states are represented by the vector $(W, W_i, S)$, where $W$ is the window size, $W_i$ is the window threshold, and $S$ is one of TCP states (N, I, T, F). Note that $W, W_i \leq W_m$, where $W_m$ is the maximum receive window size. The normal state models the way the window size changes (without packet drops) for both slow start and congestion avoidance. When a loss event occurs, each normal state can transit to one of two loss states: timeout or fast retransmit. A transition to the idle state models the time where a connection has no data to send.

From normal state to normal state, $\lambda_{nl}$ denotes its transition rate. Transition rates of Normal to timeout and normal to fast retransmit are represented as $\lambda_{nT}$ and $\lambda_{nl}$ respectively. The duration of time spent in the fast retransmit and timeout states determines the rate of departure from these states, $\lambda_{fg}$ and $\lambda_{gT}$ respectively. The duration of time that the source is idle determines the idle rate $\lambda_{I}$. The time the source is active and transmitting is defined by the active rate $\lambda_{A}$.
When a TCP source is active and there is no packet loss, congestion avoidance and slow start algorithms determine how the window size increases and therefore the window transition rate. In congestion avoidance we approximate the time between window bursts to be one RTT, denoted by $R$. This is the time from when the first packet in the window burst is transmitted to the time the acknowledgment (ACK) for this packet is received. The transition rate within one window is simply the inverse of this time, denoted by $1/R$. The probability that the window size increases depends on packet loss. For the window size to increase all packets from the current window burst must be transmitted successfully. The probability that the window size increases is therefore the probability that no packets are dropped from the burst and is denoted by $P_{ns}$.

A RTT is made up of the physical link delay, the mean queuing delay and the service time at the bottleneck link. We assume that the ACK service times are insignificant compared to the physical link delay since ACK packets are small. The duration of time between slow start states is the same as in congestion avoidance except that the window size doubles rather than incrementing by one. The transition rates are therefore the same for both slow start and congestion avoidance.

Figure 1 summarizes state transitions and transition rates of continuous time Markov Chains for TCP.

![Diagram of state transitions](image)

<table>
<thead>
<tr>
<th>Transition</th>
<th>conditions</th>
<th>Original State</th>
<th>New State</th>
<th>Transition Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$W &lt; W_{s}/2$</td>
<td>$(W, W_s, N)$</td>
<td>$(W, W_s, F)$</td>
<td>$\lambda_{ns}$</td>
</tr>
<tr>
<td>2</td>
<td>$W_{s}/2 \leq W &lt; W_{t}$</td>
<td>$(W, W_s, N)$</td>
<td>$(W, W_s, F)$</td>
<td>$\lambda_{ns}$</td>
</tr>
<tr>
<td>3</td>
<td>$W_{t} \leq W &lt; W_{m}$</td>
<td>$(W, W_t, N)$</td>
<td>$(W + 1, W_t, N)$</td>
<td>$\lambda_{ns}$</td>
</tr>
<tr>
<td>4</td>
<td>$W \leq W_{m}$</td>
<td>$(W, W_m, N)$</td>
<td>$(W, W_m, T)$</td>
<td>$\lambda_{ns}$</td>
</tr>
</tbody>
</table>

In the following, we approximate the expected transition rates for fast retransmit and timeout. We assume it takes approximately one RTT to detect a fast retransmit loss and one RTT to recover each of the packets lost. For smaller window sizes it may take an extra RTT to detect the loss but we can ignore this because the effect is minimal compared to the time to recover the losses. The departure rate from the fast retransmit state is then given by the following:

$$X(w) = \sum_{i=1}^{w} i \cdot P_{i}(w)$$

$$\lambda_{t} = \frac{1}{R \cdot X(w)}$$

where $X(w)$ is the expected number of packets lost. If $X(w)$ packets are dropped then $w - X(w)$ ACKs are received, each of which transmits a packet. Each of the lost packets is also retransmitted, and the total number of packets transmitted is $w$. Note that the probability $P_{i}(w)$ is the probability of $i$ losses out of a window of $w$ packets, assuming that packet loss is independent.

Figure 1: Continuous Time Markov Chain Model for TCP

We will calculate transition rates and derive system throughput based on the following model.

- $P_{c}$ is the drop probability due to congestion
- $P_{w}$ is the drop probability due to wireless or satellite channel loss
- $P = P_{c} + P_{w}$, the packet drop probability
- $R$, the average round trip time

$$\lambda(P, R) = \sum_{w=1}^{\infty} \sum_{i=1}^{w} \pi(w, w, N) \pi + \sum_{i=1}^{w} \pi(w, w, G_{a} \lambda_{f}) + \sum_{i=1}^{w} \pi(w, w, G_{r} \lambda_{f})$$

$$\pi = \left\{ \begin{array}{ll}
\pi_{w,w,N} & & & & \\
\pi_{w,w,F} & & & & \\
\pi_{w,w,I} & & & & \\
\pi_{w,w,T} & & & & \\
\end{array} \right\}$$

$\pi_{w,w,N}$, the distribution of a TCP active states.
$\pi_{w,w,F}$, the distribution of a TCP fast retransmit states.
$\pi_{w,w,I}$, the distribution of a TCP timeout states.

In the following, we approximate the expected transition rates for fast retransmit and timeout. We assume it takes approximately one RTT to detect a fast retransmit loss and one RTT to recover each of the packets lost. For smaller window sizes it may take an extra RTT to detect the loss but we can ignore this because the effect is minimal compared to the time to recover the losses. The departure rate from the fast retransmit state is then given by the following:

$$X(w) = \sum_{i=1}^{w} i \cdot P_{i}(w)$$

$$\lambda_{t} = \frac{1}{R \cdot X(w)}$$

where $X(w)$ is the expected number of packets lost. If $X(w)$ packets are dropped then $w - X(w)$ ACKs are received, each of which transmits a packet. Each of the lost packets is also retransmitted, and the total number of packets transmitted is $w$. Note that the probability $P_{i}(w)$ is the probability of $i$ losses out of a window of $w$ packets, assuming that packet loss is independent.
TCP may generate an immediate acknowledgment (a duplicate ACK) when an out-of-order segment is received. The purpose of this duplicate ACK is to let the other end know that a segment was received out of order, and to tell it what sequence number is expected. Since TCP does not know whether a duplicate ACK is caused by a lost segment or just a reordering of segments, it waits for a small number of duplicate ACKs to be received. It is assumed that if there is just a reordering of the segments, there will be only one or two duplicate ACKs before the reordered segment is processed, which will then generate a new ACK. If three or more duplicate ACKs are received in a row, it is a strong indication that a segment has been lost. TCP then performs a retransmission of what appears to be the missing segment, without waiting for a retransmission timer to expire. If there is not enough duplicate ACKs to trigger a fast retransmit then a timeout will occur when the retransmission timer expires. The duration of the retransmission timeout is defined in [6] as

\[ \lambda_{lt} = \frac{1}{5R} \]

In New Reno, if less than three duplicate ACKs are received, a timeout will result. We therefore assume that the probability of a timeout is the probability that less than three packets are successfully transmitted in a burst. Furthermore, the timeout probability is given when at least one packet has been dropped,

\[ P_{lt} = \sum_{x=W-2}^{Wn} P_w(x) \]

The probability of fast retransmit \( P_{lf} \) is determined as

\[ P_{lf} = 1 - P_{nl} - P_{lt} \]

where \( P_{nl} \) is the probability that no packet is dropped from the burst.

Note that the rates, \( \lambda_{nl} \), \( \lambda_{lf} \), and \( \lambda_{lt} \) are all a function of window size \( W \). The transition rates are now summarized as follows:

\[
\begin{align*}
P_w(i) &= \binom{w}{i} p^i (1-p)^{w-i}, \quad \lambda_{nl} = \frac{(1-p)^w}{R} \\
\lambda_{lf} &= \frac{1}{R \sum_{x=1}^{w} x \cdot P_w(x)}, \quad \lambda_{lt} = \frac{1}{R + 4 \cdot T_U} \\
\lambda_{ntd} &= \sum_{x=Wm-2}^{Wn} P_w(x), \quad 1- (1-p)^w - \sum_{x=Wm-2}^{Wn} P_w(x) \\
\lambda_{ntd} &= \frac{R}{R} \quad \lambda_{ntd} = \frac{R}{R}
\end{align*}
\]

Now that the transition probabilities and rates are defined, the stationary distribution, \( \pi \) of the Markov process \( \{U_k\} \) can be derived. The probability distribution of congestion windows is verified with simulation result from [6]. The following chart shows that the model matches simulation result closely. Additionally, the throughput can be calculated based on the number of packets transmitted at each state and the duration of the transition.

![Figure 2: Congestion Window Distribution for TCP New Reno](image)

When multiple packets are lost within a window, selective acknowledgments used by TCP-SNACK become valuable. TCP-SNACK makes one key change to the TCP-NewReno protocol: TCP-SNACK allows ACKs to carry information about which packet they are acknowledging. The information about which packet is dropped allows the sender to fast retransmit as long as one packet from the window gets transmitted to the receiver and none of the retransmitted packets are lost. As described previously, TCP-NewReno transitions to a timeout even when only one or two packets are lost, if the window size is small. In any case where a larger number of packets are lost, TCP-NewReno will be forced to timeout and transit to the appropriate timeout state. Therefore, the structure of the Markov chain for TCP-SNACK remains the same as the Markov chain for TCP-NewReno, and only the transition intensities differ as follows. Whereas TCP-NewReno transitioned into the fast retransmit states only upon a loss of up to three packets in a given window, TCP-SNACK will transition to fast retransmit as long as three duplicate ACKs are received for the first lost packet and none of the retransmitted packets are lost. Assuming independent packet losses, we calculate the timeout probability for TCP-SNACK and the probability for
fast retransmit/fast recovery by modifying the transitions of TCP-NewReno as follows:

\[
\lambda_{\text{red}} = \frac{1 - \sum_{i=1}^{N} p_i (x) (1 - (1 - P) x) - \sum_{i=W+2}^{W} p_i (x) (1 - P)^i}{R}
\]

\[
\lambda_{\text{fd}} = \frac{-\sum_{i=1}^{W} p_i (x) (1 - (1 - P) x) + \sum_{i=W+2}^{W} p_i (x)}{R}
\]

In tactical networks some wireless (radio) links can suffer from blockage due to terrain and mobility. Identification of blockage and separation of blockage from congestion allows a more effective congestion control policy. Radio blockage can vary depending on the terrain (urban, suburban, foliage, etc.). In urban areas blockage can result from high buildings that obstruct point-to-point radio links. A satellite beam with low elevation angle can suffer from blockage as well. Blockage periods in urban areas can sometimes last for minutes (a vehicle moving through a tunnel). In suburban areas, small buildings can cause blockage that last for shorter periods. Foliage can have an effect on the quality of radio signals as well.

Here we describe how to apply the design of a sender based TCP error discriminator that uses additional mechanism like ECN to discriminate between error types. The idea is to give TCP-SNACK sender the ability to discriminate between wireless and congestion errors by using feedback from Active Queue Management (AQM) mechanism at the middle nodes that can mark packets when congestion is happening or about to happen. If RED indicates that congestion is taking place then the error is considered a congestion error and TCP reduces its transmission rate. On the other hand, if RED did not mark packets because there is no congestion then the error is considered to be wireless error and the sender resends the lost packet without the need to reduce its transmission rate. Before we describe the work of the error discriminator, we will put some assumptions:

- In order for this mechanism to work there should be no congestion drops for ECN packet.
- In case of RED there will be no dropping. Instead, RED will send ECN whenever it expects that a congestion is about to occur. Dropping due to RED is required at the bottleneck only.

- If the RED queue is full then ECN should be sent to the TCP sender.
- If there are congestion errors as well as wireless errors then TCP should solve the congestion first by dropping transmission rate (congestion is given higher priority).

We now modify TCP-SNACK so it can discriminate between congestion and wireless errors based on the ECN feedback from RED. The changes are only made to the TCP sender:

- If a packet is received by the TCP sender, the discrimination module will read it first. If the received packet is a duplicate acknowledgment and marked by RED with an ECN or a marked packet has been received recently then we consider it an indication of a congestion error. In this case, the control is passed to TCP-SNACK congestion control mechanism.
- If a duplicate acknowledgment is received but with no marked packets recently, then it is considered as a wireless error. Retransmit the lost packet and reset the retransmission timeout timer.
- In case of timeout, pass the control to the TCP-SNACK congestion control.
- If a new acknowledgment is received with an ECN mark, then TCP reduces its transmission rate to prevent expected congestion.

3. ANALYTICAL RESULTS

The analytical result in this section is obtained through probabilistic model checker tool [8]. The main areas of interest would be the throughput comparison, link utilization, congestion window modifications, and number of packets sent within a given time.

TCP-NewReno treats all packet losses as congestion by default; it is right to treat it this way for most cases in terrestrial networks. Thus, even when an error occurs, TCP-NewReno assumes it is because of congestion and starts reducing its congestion window, which is the exact opposite of what it should be doing. Rather than reducing the amount of data sent, it should immediately retransmit all the data to recover from the error as soon as possible. The scenarios are listed in Table 2 and throughput comparison is shown in Figure 3.
Table 2: Parameters for various tactical scenarios

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Dropping Probability</th>
<th>Round Trip Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0.1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0.01</td>
<td>0.5</td>
</tr>
<tr>
<td>5</td>
<td>0.01</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0.01</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>0.001</td>
<td>0.5</td>
</tr>
<tr>
<td>8</td>
<td>0.001</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>0.001</td>
<td>2</td>
</tr>
</tbody>
</table>

TCP-NewReno and TCP-SNACK cannot differentiate packet loss from congestion and blockage. In space environments where most of the loss occurs due to blockage, this has a very bad effect because the receiver is waiting for data to be retransmitted while the sender delays sending the data. The TCP-SNACK (with or without ECN) is compared with the TCP-NewReno and various scenarios. The test scenarios are specified in Table 3.

The TCP-SNACK algorithm has a better throughput than that of the TCP-NewReno because it retransmits immediately for packet loss due to blockage. We compare the throughput of TCP-NewReno, TCP-SNACK, and TCP-SNACK with ECN for scenarios depicted in Table 3. The analytical result of Figure 4 shows that TCP-SNACK with ECN outperforms TCP-NewReno and TCP-SNACK for all scenarios. This indicates that TCP-SNACK would become a more effective method with ECN assuming that congestion is accurately indicated by ECN. The estimation accuracy of ECN will be considered in future papers.

Table 3: Parameters for various scenarios

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Pc</th>
<th>Pw</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.05</td>
<td>0.001</td>
</tr>
<tr>
<td>2</td>
<td>0.05</td>
<td>0.01</td>
</tr>
<tr>
<td>3</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>4</td>
<td>0.05</td>
<td>0.1</td>
</tr>
<tr>
<td>5</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Figure 4: TCP throughput comparisons for various scenarios

4. CONCLUSIONS

We note that a continuous-time Markov chain is developed to model TCP. To eliminate the assumption that the transition time between states is exponentially distributed, which may be difficult to justify in real TCP transactions, a discrete-time Markov chain model should be given due consideration [5, 11]. The Poisson processes used in this paper adequately model certain session arrivals such as FTP and TELNET but not others such as HTTP, SMTP and NNTP. This imposes limitations on the use of this methodology. We recommend consideration of traffic models like Markov modulated Poisson process [7] to eliminate the underlying traffic model restrictions. The error model that we used in the analysis is a random error model, but this does not faithfully resemble the actual error pattern that occurs on a real satellite channel. Future work should consider different loss models, laboratory measurement and simulation to validate these analytic models for tactical networks.

In this work we have shown the analysis of performance improvement to TCP-NewReno from incorporating PEP and ECN over satellite. The analysis suggests that TCP uses ECN feedback from congested nodes to discriminate between congestion and wireless errors and act differently in response to each type of errors. In future work, the TCP-NewReno and TCP-SNACK options modeled are particularly beneficial in cases where we need to
validate performance enhancement from PEP and ECN as well as other proposed enhancements from COTS providers. The effort of current models can also be extended to assess PEP performance when pairing PEP products with variants of TCP congestion control algorithms.

The analytical results from the model showed that TCP-SNACK outperforms TCP-NewReno in networks suffering from different wireless error rates and low or moderate congestion. The initial observations showed that discriminating between error types in case of high congestion is not helpful because in case of high congestion the priority is to resolve the congestion by reducing the sender transmission rate. Furthermore, as shown in [10], advanced measurement mechanism can provide higher accuracy of estimation to differentiate congestion loss from blockage loss, the analytical model of TCP congestion control algorithm could be revised to reflect algorithm changes by incorporating advance measurement from hosts and routers.

Another future work under study is to extend these models to assess performance of Ad-hoc TCP [9] for MANET.

5. REFERENCES


