The Impatient variant of TCP NewReno responds to partial ACKs more aggressively than the Slow-but-Steady variant. Because the Impatient variant invokes Slow Start, its loss recovery time depends on the pattern of packet drops. In this paper, we analyze the average loss recovery time of the Impatient variant of NewReno.

Introduction: There are two variants of TCP NewReno: the Slow-but-Steady variant and the Impatient variant [1]. The Slow-but-Steady variant recovers one packet per RTT (Round Trip Time), and its recovery time increases linearly in proportion to the number of lost packets. On the other hand, the Impatient variant invokes Slow Start when a large number of packets have been dropped from a single congestion window.

In this letter, we analyze the average recovery time of the Impatient variant of NewReno. Previously, Joo and Bahk [2] analyzed the packet recovery time of the Slow-but-Steady variant. When many packets are lost from a single window, we expect that the Impatient variant has shorter recovery time than the Slow-but-Steady variant. However, the recovery time depends on the pattern of packet losses and this makes the analysis difficult.

Analysis: We decompose the loss recovery behavior of the Impatient variant into
three aspects: the initial Fast Recovery phase, Slow Start phase, and Congestion Avoidance phase. The Impatient variant behaves similar to the Slow-but-Steady variant in the Fast Recovery. However, contrary to the Slow-but-Steady variant that resets the retransmit timer when it receives a partial ACK, the Impatient variant does not reset the retransmit timer until a full ACK is received. As a result, the retransmit timer expires and Slow Start is invoked when a large number of packets have been dropped. And the sender enters into Congestion Avoidance when congestion window size becomes larger than $ssthresh$ (Slow Start Threshold).

We adopted the notion of "rounds" [3] in our analysis. A round starts when the sender begins the transmission of a window of packets and ends when the sender receives an ACK for one or more these packets. Let the first round start when a sender receives the first duplicate ACK, and let $W$ be the congestion window size at that time. We assume that $d$ packets out of $W$ packets are lost. Because TCP always fully utilizes the given congestion window, the very first packet of the window is one of $d$ lost packets. Assuming that $(d-1)$ lost packets are distributed uniformly over $(W-1)$ packets, the loss probability of each packet, $p$, is

$$p = \frac{d - 1}{W - 1}$$  \hspace{1cm} (1)

Let the retransmit timer expire in the $m$-th round. The Impatient data sender recovers one lost packet per RTT until the retransmit timeout. Therefore, if less than $m$ packets have been dropped, there is no difference between the Impatien variant and the Slow-but-Steady variant.

We model the data sender’s behavior during Slow Start. During Slow Start, the congestion window increases by one segment for each arrival of a partial ACK. Because a partial ACK is generated when a lost packet is recovered, the number of partial ACKs arrived in the $n$-th round is same as the number of packets recovered in the $(n-1)$-th round. Assuming no packet loss during loss recovery, $W_n$, the
congestion window size at the $n$-th round is determined as

$$W_n = \begin{cases} 1, & n = m \\ W_{n-1} + L_{n-1}, & n \geq m + 1 \end{cases}$$  \hspace{1cm} (2)$$

where $L_{n-1}$ is the number of recovered packets in the $(n-1)$-th round.

To derive the expected value of $W_n$, we should consider the window inflation process. Window inflations are mostly caused by the packets transmitted before the retransmit timeout. Therefore, we should examine the packets transmitted during the $(m-1)$-th round. The sender transmits $(N = 1 + \max(0, \frac{W}{2} - d + m - 2))$ packets in the $(m-1)$-th round [2]. The first of $N$ packets is a retransmitted packet that recovers a lost packet and the other $(N-1)$ packets. As a result, $N$ identical ACKs are received in the $m$-th round. These $N$ identical ACKs have different effects on the loss recovery process. Let us explain the different effects with Figure which shows the loss recovery behavior in the $m$-th round. Let $k(1 \leq k \leq N)$ be the number of ACKs received before the retransmit timeout. These $k$ packets trigger the retransmission of one lost packet, and the transmission of $k$ new packets. After the timeout, $(N-k)$ more duplicate ACKs arrive. The Impatient variant ignores first two duplicate ACKs. However, third and later duplicated ACKs cause window inflation. Let $I_m$ be the number of packets transmitted in the $m$-th round due to the window inflation. Then, $I_m$ is

$$I_m = \begin{cases} 0, & N - k \leq 2 \\ 2, & N - k = 3 \\ N - k, & N - k \geq 4 \end{cases}$$  \hspace{1cm} (3)$$

If $I_m$ is zero, the $m$-th round recovers only one packet and the packets transmitted in the $m$-th round generate identical ACKs. The data sender transmits $(k+2)$ packets in the $m$-th round, that is, $(k+1)$ packets before the timeout and one packet when the timeout. Because $(k+2)$ identical ACKs are received in the $(m+1)$-th round, we have
\[ I_{m+1} = \begin{cases} 
2, & \text{if } k = 2, I_m = 0 \\
1 + k, & \text{if } k \geq 3, I_m = 0 \\
0, & \text{otherwise}
\end{cases} \] (4)

We assumed \( I_n \) is zero when \( n \) is larger than \((m+1)\). It is generally true because the effect of window inflation is considerable only when many duplicate ACKs are received. For the simplicity of analysis, we ignore small window inflation.

Note that the first packet sent in the \( n \)-th round is a lost packet and other lost packets randomly distributed among the other retransmitted packets. Because we assumed that the packet loss probability is \( p \), the average number of recovered packets in the \( n \)-th round, \( E(L_n) \), is

\[ E(L_n) = \begin{cases} 
1 + p(E(W_n) - 1 + E(I_n)), & n = m, m + 1 \\
1 + p(E(W_n) - 1), & n \geq m + 2
\end{cases} \] (5)

From (2) and (5), we have

\[ E(W_n) = \begin{cases} 
1, & n = m \\
2 + pE(I_m), & n = m + 1 \\
(2 + p + p(1 + p)E(I_m) + pE(I_{m+1})) + \frac{1}{p}(1 + p)^{n-m-2} - \frac{1-p}{p}, & n \geq m + 2
\end{cases} \] (6)

Now, let us compute, \( S_n \), the number of packets recovered during the first \( n \) rounds.

From the first round to the \((m-1)\)-th round, one packet is recovered in each round.

And, \( L_i \) packets are recovered in the \( i \)-th round where \( i \geq m \). Therefore, \( E(S_i) \) is

\[ E(S_n) = \begin{cases} 
n, & n \leq m - 1 \\
m + pE(I_m), & n = m \\
m - \frac{1-p}{p} + (2 + p + p(1 + p)E(I_m) + pE(I_{m+1})) + \frac{1}{p}(1 + p)^{n-m-1}, & n \geq m + 1
\end{cases} \] (7)

We are interested in the average time to recover \( d \) lost packets. Suppose it takes \( x \) rounds on average to recover \( d \) packet losses, then we have

\[ E(S_x) = d \] (8)
Finally, we should consider Congestion Avoidance process to compute the recovery time exactly, when $W_n$ is larger than $sthresh$. If the congestion window increases as large as $sthresh$, the sender enters into Congestion Avoidance phase. Let $x_{SS}$ be the number of rounds when the congestion window grows to $sthresh$. $x_{SS}$ is obtained from the equation $W_{x_{SS}} = stthresh$. Only when $x$ obtained from (8) is larger than $x_{SS}$, we should adjust the recovery time due to the Congestion Avoidance. In the Congestion Avoidance phase, the congestion window increases slowly. Because most packets are recovered before the congestion window size grows to $sthresh$ and the congestion window size growth rate in the Congestion Avoidance phase is small, we assume that the congestion window size is fixed to $sthresh$. Then, during Congestion Avoidance, $sthresh$ packets are transmitted per RTT. However, the last round where the recovery finishes requires more detailed analysis. Note that some packets among $sthresh$ packets can exceed the loss interval in the last round. Taking an average of the case where one packet is recovered and the case where $sthresh$ packets are recovered in the last round, the average recovery time, $x$, is given by

$$x = x_{SS} + \frac{1}{2} \left( \frac{d - E(S_{x_{SS}})}{1 + p(stthresh - 1)} + \frac{d - E(S_{x_{SS}}) - 1}{p(stthresh - 1)} + 1 \right)$$

(9)

We can obtain the average recovery time, $x$, from (8) and (9).

**Results:** We examine the correctness of the analysis by compare the analysis results with simulation results. Figure shows the recovery time as a function of the number of lost packets ($d$). We fix $W$ to 30 and $m$ to 4 in this experiment. Because we are interested in the average recovery time, we generated one hundred random packet loss patterns and compute the average to obtain the simulation result. In figure, we can observe that the analysis result agrees with the experiment result in all range.
Conclusions: The Impatient variant of TCP NewReno could reduce the packet recovery time when many packets are lost in one congestion window. We analyzed the average loss recovery time of the Impatient variant of NewReno. Our performance study showed that the analysis correctly measure the average performance of the Impatient variant.

References:

Author’s affiliations:
Sunwoong Choi and Chongkwon Kim(School of Computer Science and Engineering, Seoul National University, San 56-1, Sillimdong, Kwanakgu, Seoul, Korea) {schoi, ckim}@popeye.snu.ac.kr

Figure captions:
Fig. 1 Recovery Behavior in the m-th round
Fig. 2 Analysis and Experiment Result
Figure 1
Figure 2

![Graph showing the relationship between Number of Lost Packets and Recovery Time (RTT). The graph includes data points for Simulation and Analysis, with a trend line indicating an increase in recovery time as the number of lost packets increases.]