A Parameterized Model of TCP Slow Start

Xiaoheng Deng, Zhigang Chen, Lianming Zhang

School of Info Sci & Tech, Central South University, Changsha, 410083, China
dengxh@hunnu.edu.cn, czg@csu.edu.cn, zlm@hunnu.edu.cn

Abstract. Based on analysis on multiple packet losses of standard slow start caused by exponential growth of congestion window (*cwnd*), this paper proposes a new phase-divided TCP start scheme and designs a parameterized model to reduce packet losses and improve TCP performance. This scheme employs different of *cwnd* growth rules while *cwnd* is under and over the value of half window threshold (*ssthresh*) respectively, namely exponential growth and negatively exponential growth, which greatly decreases probability of multiple packet losses from a window of data and guarantees that a connection smoothly joins the Internet and transforms into congestion avoidance. Parameterized model adjusts the duration of slow start and acceleration of increasing *cwnd* to improve performance of slow start phase through various parameter setting. An adaptive paraeter setting method is designed. And the simulation results show that this new method significantly decreaseds packet losses and improves the stability of TCP and performance of slow start, and also achieves good fairness and friendliness to other TCP connections.

1 Introduction

With expanding of the Internet applications and scale, TCP is widely deployed, providing reliable end-to-end Internet services. Statistic data show that 95 percent of data flows belong to TCP on the Internet [1]. Since TCP was produced, researchers have done many work on it and proposed several enhanced variants [2,3,4]. TCP adopts slide window mechanism to control network congestion and slow start takes effect while a session starts, the sender opens one segment size congestion window and exponentially increases *cwnd* until *cwnd* reaches a threshold, *ssthresh*, therefore, slow start effectively avoids bursty traffic while new connections join network.

Connections are classified into long-lived and short-lived connections according to the duration. Network measurement [1] shows that short-lived flows, such as WEB and TELNET, are the majority, and long-lived flows, like FTP, are the minority but transfer most data packets of TCP flows. Short-lived flows often end before they come to steady state; namely, they often locate at startup phase. Obviously, the performance of slow start directly impacts on the transmission utility of short-lived flows. In practice, short-lived flows convey the data of important Internet services, which require high bandwidth and short delay. In the same way, slow start also takes effect while long-lived flows start and a retransmission timeout takes place, thus impacts on performance of long-lived flows. Although slow start's duration is very short, it is of great significance to improve performance of data transmission. The

H. Jin et al. (Eds.): NPC 2004, LNCS 3222, pp. 316-324, 2004.

congestion window of standard slow start doubles with *RTT*, shown in figure 2 and figure 3, the equation follows as (1),

$$cwnd(t+T) = \begin{cases} 2 \times cwnd(t), if\ cwnd(t) < ssthresh \\ ssthresh, \quad if\ cwnd(t) > ssthresh \end{cases} \quad (1)$$

Researchers have proposed many enhanced schemes to improve slow start, M. Allman [5] has set a larger initial window(LIW) to improve the performance; TCP Fast-start [6] records network parameters of network recently to reduce the start time of a new connection, decrease short abrupt transmission delay, and maintain high utility while network keeps steady. SPAND [7] picks up current network state and gains optimal initial parameters. J. Hoe's method replaces the default setting of *ssthresh* with an estimated value to ensure that *cwnd* reaches an appropriate value [8]. That recent history information is used to initialize parameters of new connections has been presented in [9]. TCP Vegas [10] restricts the exponential window growth, and doubles *cwnd* every other *RTT*. The above approaches partially optimize slow start, but each still has weakness. TCP Fast-start has strict condition of steady network. J. Hoe's method is problematic in practice. Parameter setting based history information violates slow start principle and cannot fit dynamic change of network. TCP Vegas cannot avoid multiple packet losses in one window.

Considering the limitations, we propose a new phase-divided and gradually approaching slow start algorithm, called P-Start. P-Start employs standard slow start mechanism and *cwnd* grows exponentially while congestion window is less than *ssthresh*/2; If *cwnd* is equal or greater than *ssthresh*/2, the congestion window, $cwnd_{i+1}$, is not directly set with *ssthresh* but only increases (*ssthresh*-$cwnd_i$)/2, and iterates until (*ssthresh*-$cwnd_i$) is less than the factor of δ that lies from 2 to *ssthresh*/2. This approach combines exponential and negatively exponential growth, of which congestion window gradually approaches *ssthresh*, to improve stability of TCP and decrease multiple packet losses. The rest of the paper is organized as follows. Section 2 proposes phase-divided and gradually approaching slow start algorithm and the parameterized model; Section 3 validates the algorithm through a series of simulating experiments; Section 4 gives the conclusion and points out further research direction.

2 P-Start

2.1 Motivation of P-Start

TCP slow start probes network bandwidth, and its inherent property of exponential growth of window from one packet results in the following problems: first, congestion window starts with one packet and spends many *RTT*s, which brings low utility of short-lived connections. So researchers propose a larger initial window and adopts fast start [5,6] to reduce the duration of slow start and improve network utility. Second, the source nodes are blind to the available bandwidth and use the default initial *ssthresh*. The exponential growth of congestion window results in severe overflow of the buffer of the bottleneck link and multiple packet losses, which makes self-clock loss and retransmission timeout of TCP. And retransmission timeout causes

global synchronization, which greatly degrades network utility and brings oscillation of queuing delay. TCP restarts and regains self-clocking.

Efficient measurement technology, which is used to probe available network bandwidth, adaptively set slow start *ssthresh*, and eliminate the limitation of static parameters setting, is potential method. For dynamics of data flows and delay of system feedback, even though, effective bandwidth measurement cannot gain entire match between *ssthresh* and available bandwidth. In the last *RTT*, the send window increase near *ssthresh*/2, the largest increment, but the sending rate is close to network's capability. The over increment of window causes multiple packet losses.

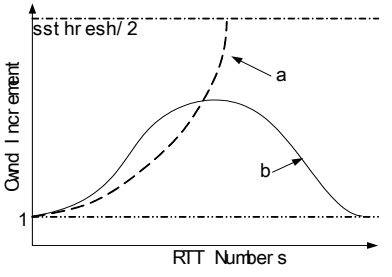

Fig. 1. *Cwnd* acceleration

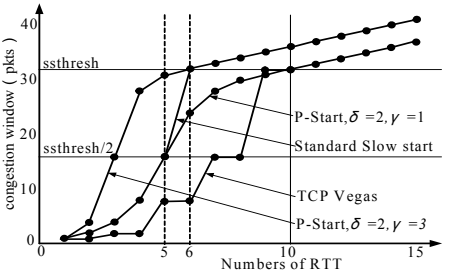

Fig. 2. Comparison of startup mechanisms

The growth of window of standard slow start is shown as curve a in figure 1; the increment of window becomes larger and larger. In fact, while a connection starts, it should gradually increase congestion window from a small one to avoid large initial window, which brings bursty traffic and causes network congestion. When sending rate is near network capacity, the increment should be reduced to smoothly transform into congestion avoidance, which is shown as curve b in figure 1.

2.2 Elements of P-Start

The key idea of P-Start is that congestion window increases exponentially while it is less than *ssthresh*/2, otherwise, increases (*ssthresh-cwnd*)/2 and gradually approaches *ssthresh* until the (*ssthresh-cwnd*) is less than the factor of δ (ssthresh/2≥ δ ≥2), and then *cwnd* is set with *ssthresh* and transforms into congestion avoidance phase. In contrast to TCP Vegas, P-Start has the same duration when the factor is set with 2, the low bound. It means that the longest process of P-Start is same as long as TCP Vegas's, but P-Start can efficiently decrease probability of multiple packet losses for *cwnd* increment of P-Start becomes smaller and locates middle phase of startup. And *cwnd* can be represented as (2), shown in figure 2 and figure 4.

$$cwnd(t+T)=\begin{cases}2\times cwnd(t), & if\ cwnd < ssthresh/2\\ (cwnd(t)+ssthresh)/2, & if\ ssthresh-cwnd\ge\delta\\ & and\ ssthresh > cwnd \ge ssthresh/2\\ ssthresh, & else\end{cases}\qquad (2)$$

The major feature of P-Start is that *cwnd* increases with a small amplitude at start phase and transition phase to congestion avoidance phase, sending rate changes

smoothly with little impact on other shared connections, and maintains network stability, decreases oscillation. Algorithm is shown as following,

```
1. init,cwnd=1,ssthresh=ssth_init,reset δ;
2. send cwnd packets;
4. if cwnd<ssthresh/2,then cwnd=2*cwnd,  goto 2;
5  if cwnd≥ssthresh/2 and ssthresh-cwnd>δ then
   cwnd=(cwnd+ssthresh)/2, goto 2;
5. else cwnd=ssthresh, goto 5;
6. if received 3 dup-ACKs or retransmision time out,
   then re-enter slow start phase, goto 1;
7. enter congestion avoidance phase;
```

We compare the window change between standard slow start [12] and P-Start. First, we assume that all segments are successfully acked and round trip time is fix. Let $N+1=\log_2^{ssthresh}$, So the duration is equal to $T_{ss}=(N+1)\times RTT$ according to slow start elements. Increment of window are shown in figure 3, in the first N-1 *RTT*s, the total increment is *ssthresh*/4; In the last two *RTT*s, *cwnd* increases *ssthresh*/4 and *ssthresh*/2 respectively, total increment of the last two RTT reaches 3 quarters of *ssthresh*. In high-speed network, congestion window size is very large; The bursty traffic caused by great imcrement of *cwnd* should result in great impact on network stability, packet loss of all connections sharing the bottleneck link, global synchronization and degrading of performance. P-Start is shown in figure 4, while δ =2, the largest increment of *cwnd* is equal to *ssthresh*/4 and occurs twice during slow start, and bursty traffic reduces by 50%. Congestion window changes smoothly, the utility of P-Start is lower than standard slow start's, which is shown in figure 2. But P-Start can maintain network higher utility while network congestion occurs by decreasing packet losses. P-Start is a general slow start, which can be applied in wider areas, through varying δ from 2 to *ssthresh*/2. If δ = *ssthresh*/2, P-Start turns to standard slow start. P-Start reduces its duration and improves utility if δ is set with the value greater than 2.

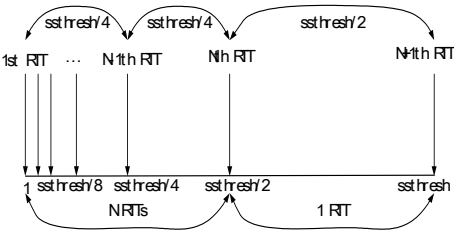

Fig. 3. Window change of slow start of TCP

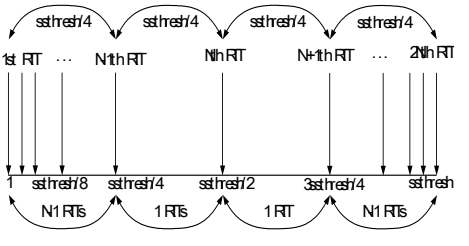

Fig. 4. Window change of P-Start (δ =2)

2.3 Flexible Parameter Setting Model

According to the statement of section 2.2, if δ =2, P-Start has the same duration as TCP Vegas's, however, P-Start diminishes the granularity of increment of *cwnd*, and significantly improves the smoothness of sending rate change, shown in figure 2; and

if δ =*ssthresh*/2, P-Start becomes standard slow start. P-Start can flexibly select the value of δ from *ssthresh*/2 to 2. Compared with standard slow start, the behaviors are same while *cwnd*<*ssthresh*/2, exponential window growth, but congestion window of P-Start is negatively exponential growth of the difference between *ssthresh* and *cwnd*, represented as (2). The duration is approximate the double of standard slow start's, which causes low performance at startup phase, shown in figure 2. So it is necessary to improve the performance and synchronously keep smooth transition to congestion avoidance. An increment factor of γ ($\gamma \geqslant 1$) is introduced, which means that *cwnd* increases γ packets if one packet is successfully acked. *Cwnd* becomes (γ +1) times of the former one during one *RTT* if all packets are successfully acked, so this factor γ represents the acceleration of *cwnd* growth. For example, if γ =1, δ =2， P-Start is the algorithm stated in section 2.2, if γ =3, δ =2, and *cwnd*<*ssthresh*/2, *cwnd* becomes 4 times as the original one, else if *cwnd* $\geqslant$*ssthresh*/2, *cwnd* increases $\frac{\gamma(ssthresh - cwnd)}{\gamma + 1}$ every *RTT*, namely 3(*ssthresh-cwnd*)/4, until (*ssthresh-cwnd*)< δ , which is shown in figure 2. So *cwnd* of P-Start can be computed as (4),

$$cwnd(t+T) = \begin{cases} (\gamma+1) \times cwnd(t), & if\ cwnd < ssthresh/2 \\ \dfrac{\gamma \times ssthresh + cwnd(t)}{\gamma+1}, & if\ cwnd \geq ssthresh/2 \\ & and\ ssthresh - cwnd \geq \delta \\ ssthresh, & if\ cwnd > ssthresh\ or\ ssthresh - cwnd < \delta \end{cases} \quad (3)$$

The performance of the above parameterized model is determined by the static parameter setting of γ and δ ,P-Start should be more adaptive to the dynamics of Internet. As large initial window has been proposed to improve the performance of slow start with the same effect of the factor δ in P-Start, so the value of δ is decided by large initial window option. For the reason of performance, startup phase cannot last too long, the time taken by various slow start mechanisms are shown in table 1.

Table 1. duration of startup phase (*RTT* is fix & all packets are successfully acked)

Slow Start	TCP Vegas	P-Start		
		γ =1, δ =2	γ =3, δ =2	Any given γ, δ
$RTT \times \log_2^{ssthresh}$	$2RTT \times \log_2^{ssthresh}$	$2RTT \times \log_2^{ssthresh/2}$	$2RTT \times \log_4^{ssthresh/2}$	$2RTT \times \log_{\gamma+1}^{(ssthresh/2-\delta)}$

Generally, the duration is no longer than 10 *RTT*s, P-Start takes time, $T_{ss} = 2RTT \times \log_{\gamma+1}^{(ssthresh/2-\delta)}$, and *ssthresh* can be gained by the method of J. Hoe [8]. If standard slow start takes time less than 5 *RTT*s, namely $\log_2^{ssthresh} \leq 5$, P-Start set γ with 1, else if $5 \leq \log_2^{ssthresh} \leq 10$, P-Start sets γ with $\alpha \times \log_2^{ssthresh/5}, \alpha \geq 1$, else γ follows equation (4). And the adaptivity of P-Start needs further research.

$$\log_{\gamma+1}^{(ssthresh/2-\delta)} \leq 5 \quad (4)$$

3 Network Simulations and Validation

To validate P-Start, we implement it in network simulator NS2 [13] in environment of red hat 9.0 linux, network topology is shown in figure 5, and simulation scenarios includes various mechanisms, such as TCP standard slow start, TCP Vegas and P-Start with different parameter setting. We design a series of experiments and compare the simulation results to figure out the advantage of P-Start.

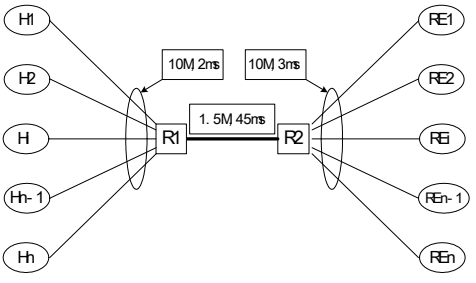

Fig. 5. Topology of network simulation

As shown in figure 5, N connections share a bottleneck link consisted of routers R1, R2, bandwidth of the bottleneck link is 1.5Mbps. Time labeled in figure 5 is one-way delay, packet size is 1024 Bytes, buffer size L is 25 packets size, and type of data transferred in network are FTP.

In experiment one, link is injected 0.5 Mbps background traffic, segment-discarding strategy of bottleneck link is Droptail. The TCP connection we measured has a fixed share of 1Mbps and 8 buffer units. According to network transmission property, product of network bandwidth and delay (BDP) can be computed as follow,

$$BDP = bandwidth \times RTT = 1.0M \times 2(2+45+3) = 12500Bytes$$

BDP is approximate 12 packets size, and the available buffer size is 8 packets size, so the pipe's capacity is 20 packets. The file size transferred is 60 Kbytes. In TCP protocol, the window of the sender is the smaller one of *cwnd* and the receiver announced window *rwnd*. *Rwnd* is set 18,20,24,36,64 packets size, of which value is set greater, equal and less than actual network bandwidth. Relation between packet losses and window size of various slow start mechanisms is shown in figure 6.

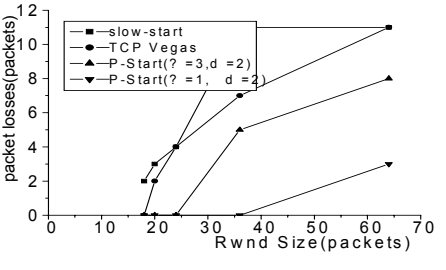

Fig. 6. Packet losses vs. rwnd

Table 2. comparison of utilities

Time	Slow start	TCP Vegas	P-Start	
			$\gamma=1, \delta=2$	$\gamma=3, \delta=2$
2 Sec	48.6%	35.1%	42.7%	69.6%
4 Sec	74.3%	78.2%	68.9%	75.6%
6 Sec	84.1%	91.3%	77.2%	85.2%
8 Sec	87.5%	93.5%	85.4%	88.3%

Throughput of slow start is of great significance to performance of services, especially to WEB services. Table 2 shows utility of slow start, the simulation results indicate that standard slow start has better performance than TCP Vegas and P-Start($\gamma=1, \delta=2$), and has worse utility than P-Start($\gamma=3, \delta=2$). P-Start gains better utility, TCP Vegas has better utility while TCP is at congestion avoidance phase. P-Start can achieve different utility responding to different parameter configuration. And more packets are dropped while P-Start combines with large initial window of 4, fewer packets are dropped and better performance is achieved while P-Start combines with J. Hoe's method. Considering the space, the detail is ignored.

In order to check the impact on network stability of slow start, we monitor the bursty traffic and instantaneous queue length of bottleneck link if the segment dropping strategy is RED(Random Early Detection). Slow start and P-Start($\gamma=1, \delta=2$) start a new FTP connection every 0.5 second and end 0.7 second later. The instantaneous queue length of bottleneck link is shown in figure 8, P-Start can effectively maintain the stability of network for the relatively smoother change of window, which causes smaller oscillation.

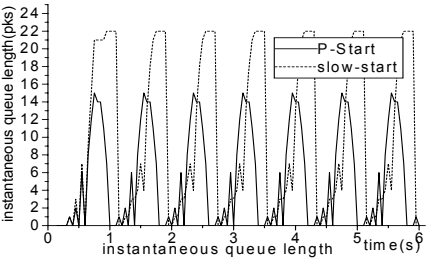

Fig. 7. Comparison of instantaneous queue length

Experiments show that P-Start can effectively decrease packet losses, lighten network oscillation, and improve performance of network with appropriate parameter setting, and the adaptive parameter setting requires further research.

Fairness reflects bandwidth allocation among various connections in the bottleneck link, we use Jain's Fairness Index represented as following [15] :

$$Fairness\ Index = \frac{(\sum_{i=1}^{n} x_i)^2}{n\sum_{i=1}^{n} x_i^2} \quad (5)$$

Where x_i is the throughput of the ith flow and n is the number of total flows. The fairness index is ratio, which lies between 0 and 1. The upper bound value of 1 shows

that all flow share the same bandwidth of bottleneck link. Simulations with different number of flows are put up to test fairness of P-Start. We calculate the Fairness Index for TCP Reno and P-Start for each simulation, and the average Fairness Index is 0.9945 and 0.9949 respectively. Simulation results show that P-Start only improves the performance of flows at the start and has no impact on other phases of flow.

4 Conclusion

This paper analyses the impact of TCP slow start on network transmission and difficulties of deployment effective and stable slow start, and proposes a new phase-divided slow start mechanism. This mechanism adjusts the rule of congestion window change, fast and smoothly transforms from slow start to congestion avoidance and decreases the damage caused by multiple packet losses. It has no effect on TCP congestion avoidance but benefits short-lived flows and flows in long fat pipes. The further research is to apply effective and accurate bandwidth measurement technology to dynamically set proper threshold value of slow start, which get over the limitation of static parameter configuration and enhance adaptivity to improve network utility.

Reference

1. K. Thompson, G. J. Miller, and R. Wilder, “Wide-Area Internet Traffic Patterns and Characteristics”, IEEE Network, Vol. 11, No. 6, pp. 10-23, November 1997
2. V. Jacobson, “Congestion Avoidance and Control”, ACM Computer Communications Review, 18(4) : 314 -329, August 1988
3. V. Jacobson, “Berkeley TCP Evolution from 4.3-Tahoe to 4.3 Reno”, Proceedings of the 18th Internet Engineering Task Force, University of British Colombia, Vancouver, BC, Sept. 1990
4. K. Fall, S. Floyd, “Simulation-based Comparisons of Tahoe, Reno, and SACK TCP”, Computer Communication Review, V. 26 N. 3, July 1996, pp. 5-21
5. M. Allman, C.Hayes, S.Ostermann, “An Evaluation of TCP with Larger InitialWindows”, ACM Computer Communication Review, Vol.28 No. 3, pp 41-52, July, 1998.
6. V. N. Padmanabhan, R. H. Katz, “TCP Fast Start: A Technique for Speeding Up Web Transfers”, Proceedings of IEEE GLOBECOM’98, Sydney, Australia, November 1998.
7. S. Seshan, M. Stemm, R. H. Katz, “SPAND: Shared Passive Network Performance Discovery”, Proceedings of USITS’97, Monterey, CA, December 1997.
8. J. Hoe, “Improving the Start-up Behavior of a Congestion Control Scheme for TCP”, Proceeding of ACM SIGCOMM’96, Stanford, CA, pp. 270-280, August 1996.
9. CHEN Jing, ZHENG Ming Chun, MENG Qiang, “A Network Congestion Control Algorithm Based HistoryConnections and Its Performance Analysis”, Journal of Computer Research and Development (in Chinese), Vol 40,No 10，Oct 2003, P1470-1475
10. L.S. Brakmo, L.L. Perterson, “TCP Vegas: End-to-End Congestion Avoidance on a Global Internet”, IEEE Journal on Selected Areas in Communication, Vol. 13, Nov. 8, October 1995
11. Claudio Casetti, Mario Gerla, Saverio Mascolo et al, “TCP Westwood: End-to-End Congestion Control for Wired/Wireless Networks”, In Wireless Networks Journal 8, 467-479, 2002
12. J. Postel, “Transmission Control Protocol, Request for Comments 793”, DDN Network Information Center, SRI International, September 1981.

13.NS project, the network simulator-ns-2 (EB/OL). http://www.isi.edu/nsnam/ns/
14.Haining Wang, Hongjie Xin, Douglas S. Reeves et al, “A Simple Refinement of Slow start of TCP Congestion”, ISCC2000, July 04-06,2000 Antibes, France
15.R. Jain, “The art of computer systems performance analysis”, John Wiley and sons, AA76.9.E94J32,1991.