Delay Performance Analysis of Token-Based Fast-TCP in Window Limit Systmes [*]

PENG Fei, CHENG Shi-duan

(*National Key Laboratory of Switching Technology and Telecommunication Networks, Beijing University of Posts & Telecommunications, Beijing* 100876, *China*)

E-mail: fpengcn@yahoo.com.cn; chsd@bupt.edu.cn

http://www.bupt.edu.cn

Received March 15, 2000; accepted July 30, 2001

Abstract: In this paper, the authors first provide a close form differentiate formula to analyze the buffer requirement for zero loss of TCP in networks with high bandwidth delay product. Then, A Token Based Fast-TCP algorithm is analyzed and modeled in WANs when the window size is smaller than Bandwidth-Delay Products (BDP) or in satellite networks where window size is not allowed to be larger than BDP because link errors. It proves that Token Based ACKs delay algorithm not only reduce buffer occupancy largely, but also gain the average same throughput as TCP with infinite buffer size as well. The authors also apply the analysis to multiple connection case that is not included in previous work.

Key words: TCP; ACK; bandwidth-delay products; differentiate formula

In Internet, a crucial issue is the high end-to-end propagation delay of connections in long haul networks. Since TCP flow control mechanism can only indirectly detect congestion by keeping track of how many packets are lost, congestion control has to be initiated after packet losses due to congestion have already happened. Therefore, if TCP control time cannot be speeded up, TCP will cause major overloads and outages on long haul networks for depending on when congestion hits, the drop of packets may come too late if the source has already sent out all the data it intended to. Furthermore, the maximum window size allowed in current systems is not large enough to catch up with Bandwidth-Delay Products, for example, in satellite networks, link bandwidth is always underutilized because window reduction are sometimes initiated by losses due to transmission errors.

For the long control loop problems, a mechanism named Fast-TCP is proposed to relieve congestion in intermediate nodes by effectively controlling ACKs flow. In this paper, we are motivated to use and analyze a robust scheme called token-base FTCP. Its prototype is shown in Fig.1. Unlike previous simple implementation of F-TCP with a fixed threshold set, ACKs are allowed to leave only if there are tokens and the calculation of Token number is based on network resource, such as spare buffer size. So, it avoids determining the rate of Delaying ACKs which is a hazard problem. for in real networks traffic changes so quickly and frequently that it is difficult to grasp.

---

[*] Supported by the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No. 20010013003 (国家教育部博士点基金)

Peng Fei was born in 1975. She received her Ph.D. degree from the Beijing University of Posts and Telecommunications in 2000. Her research focuses on Internet flow control. **Cheng Shi-duan** was born in 1940. She is a professor and doctoral supervisor at the Beijing University of Posts and Telecommunications. Her research interests cover ISDN, ATM, TCP/IP, switching software, protocol engineering, traffic engineering, network performance, QoS, security and survivability.

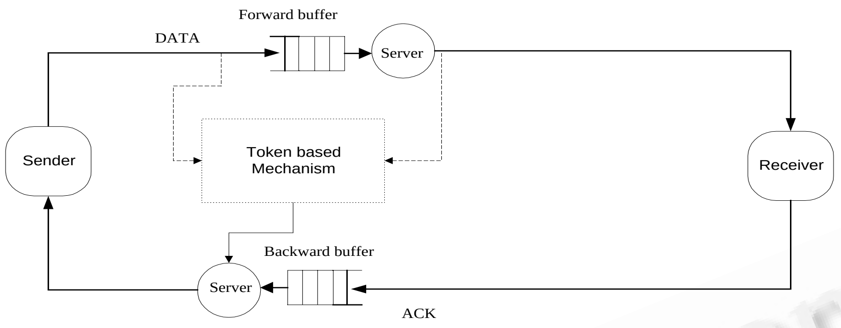

Fig.1 System model for token fast-TCP

The algorithm contains three parameters: token buffer capacity: B_T; ACKs buffer capacity: B_A; number of tokens: N_T. When the condition of data packets arriving or leaving the resource changes, this mechanism collects this information and calculates number of tokens. Some temporary variables are needed: average packet length: *Lp*; data packet counts: *Pc*.

We give an example of implementation of F-TCP. The procedure of this implementation can be depicted with the pseudo-code:

```
Initialization: Packet count=0; $B_T$=$N_T$=Forward buffer capacity/Default packet length, then
If a data packet arrives at the forward buffer,
Recalculate the average packet length $Lp$ and $B_T$
    packet count = packet count+1;
    $Lp$=($Lp$+input packet length)/packet count
    $B_T$=Forward buffer capacity/Lp
Calculate NT
    $N_T$ =Spare Forward buffer capacity/Lp
    $N_T$ =min($B_T$, $N_T$);
End
If a data packet leaves the forward buffer,
    Calculate $N_T$
    $N_T$ =Spare Forward buffer capacity/Lp
    $N_T$ =min($B_T$, $N_T$);
End
If $N_T$>0
    an ACK is served,
    $N_T$ =$N_T$ −1;
End
```

In this paper, we derive expressions for TCP and Token Based F-TCP, in conditions when congestion window size is not allowed to be larger than BDP for the limit of window size option or the link error rate such as satellite networks. Section 1 describes the system model. Then, analysis for the evolution of a single connection for TCP is given in Section 2. Section 3 restricts our attention to Token Based F-TCP in which delay and buffer requirement are analyzed. Section 4 contains simulation results for single and multiple connections. Section 5 is conclusions.

1 System Model

1.1 Background on TCP

The following is a very abbreviated and over simplified description of TCP congestion control algorithm. We define an epoch of a window of a TCP connection to be the time period from the initial starting of a window to the consecutive beginning of another window. If there is no long-lived packet queue at the router in the beginning of current cycle, the duration of current window epoch is Round-Trip Time (*T*); otherwise, it is larger than *T* for the first packet in current window has to wait data in the queue of previous window to transmit out. The amount by which the congestion window increases during an epoch is important measure of how rapidly the window size is changing. In Slow-start phase, CWND doubles during an epoch. In contrast, when in congestion avoidance phase, CWND increases by approximately 1 during an epoch.

1.2 System model for token based F-TCP

The system model of Token F-TCP can also be described in Fig.1. We consider a single bottleneck link with capacity u packets per second and a FIFO buffer of size B packets. The round-trip propagation delay for a packet is denoted by τ and $T=\tau+1/u$ denotes the propagation delay plus the service time. At time t, the window size, buffer size and the number of tokens are separately denoted as $W(t)$, $q(t)$, and $N(t)$. The SSTHRESH of window is denoted as $w_{t.}$ With infinite traffic source, the units of data are maximum sized packets.

The number of tokens calculated is on the basis of spare forward buffer capacity and we add a parameter named "token threshold" denoted as Th to ensure the buffer delay ACK in advance, that is, if $N(t)>Th$ (originally here is 0) and an ACK is served, then $N(t)=N(t)-1$. ACKs leak by a normal rate of $1/u$ which is the same as the transferring of data packet in the forward path. ACKs buffer capacity of this mechanism is assumed to be large enough to avoid ACK loss.

2 Analysis Model for TCP in One Connection Case

2.1 Slow-Start phase

As have mentioned above, if there is no long-lived packet queue at the router, the duration of a window epoch is the same as T. Slow-start phase starts from $w_D=1$ with maximum window size of SSTHRESH (denoting w_i be the maximum window size of the ith cycle). The growth rate of the window is exponential in slow start phase and we denote it as $w_i=2$ ^ i ($i>1$) (see Ref.[4]).

We now take our interest in each mini-cycle of evolution of window size and buffer occupancy. The ith mini-cycle with the maximum window size w_i refers to the time interval $[iT,(i+1)T]$. According to the definition of a window epoch T, at the beginning of each window cycle, the window size $W(iT)$ at time iT is 2^{i-1} for there are totally w_{i-1} ACKs coming back from the previous window, it reaches 2^i at time t_i before the next cycle begins, that is $t_i<T$. We can unify the analysis of this phase by a well-modeled continuous-time equation to describe the dynamics of window and queue evolution. Let dW/dt denote the rate of window growth with time, dW/da the rate of window growth with arriving acknowledgments, and da/dt at which the acknowledgments are arriving, then during slow start phase, the differential equation can be expressed as:

$$\frac{\mathrm{d}W}{\mathrm{d}t}=\frac{\mathrm{d}W}{\mathrm{d}a}\frac{\mathrm{d}a}{\mathrm{d}t}, \tag{1}$$

$$\frac{\mathrm{d}W}{\mathrm{d}a}=1, \tag{2}$$

$$\frac{\mathrm{d}W}{\mathrm{d}t}=\frac{\mathrm{d}a}{\mathrm{d}t}. \tag{3}$$

An approximation to the term of W(t) vs. ACKs directly from the window adjustment algorithm. In slow start phase it is equal to 1 for one acknowledgment causes one more packet of window to send out. Since ACKs have the same space as data in the bottleneck link. As have described,

$$\frac{\mathrm{d}a}{\mathrm{d}t}=u\ . \tag{4}$$

The source has twice the sending rate as the returned ACKs and the queue would be built up at rate of u, the window size and forward queue in mini-cycle [iT,(i+1)T] evolves as follows:

$$\frac{\mathrm{d}q}{\mathrm{d}t}=2\frac{\mathrm{d}a}{\mathrm{d}t}-\frac{\mathrm{d}a}{\mathrm{d}t}=u\ , \tag{5}$$

$$W(iT+t)=W(iT)+ut,\qquad W(iT)=w_{i-1}=2^{i-1},\ t\ge 0\,, \tag{6}$$

$$q_i(t)=ut. \tag{7}$$

The time spent for window reaches the maximum size of 2^i in this cycle according to (6) is

$$t_i=\frac{2^{(i-1)}}{u}. \tag{8}$$

From Eqs.(7) and (8), the buffer size reaches 2^{i-1} which is approximately half of a window data in the ith cycle. Then, no ACK is available and the queue empties out at the rate of u:

$$\frac{\mathrm{d}q}{\mathrm{d}t}=-u\ . \tag{9}$$

Since the rate to build up the buffer is similar to that to empty it, also t_i time is cost to empty the queue. The total time spent in the buffer is twice of t_i after the start time of this cycle iT begins.

With the premise that no long-lived packets would exit in the queue after each cycle. We deduce: $2t_i<T$, that is, the time spent in the queue should be smaller than Round-trip delay T. With the maximum SSTHRESH window size in Slow-Start phase, the following limit can guarantee the same duration of each window epoch in such phase,

$$w_t<uT\ . \tag{10}$$

As buffer possesses half window data in each cycle, with the slow start SSTHRESH of w_t, we can determine the buffer requirement B for no congestion loss is,

$$B>w_t/2\,. \tag{11}$$

Starting from an initial value of one, the number of packets transmitted successfully during slow start is well approximated by the window size w_t. The time spent for the reach of w_t is $T[\log_2 w_t+1]$ for the window size of each cycle is doubled with the duration of T.

2.2 Congestion avoidance phase

After window reaches w_t, it begins congestion avoidance phase. Unlike Slow-Start phase, the window size increases by 1 in each cycle of congestion avoidance phase. There are w_t+i packets in each window cycle, i is increased from 1 to w_t. Before the window increases 1 packet at last in each cycle, the input rate of queue is equal to its output rate and the queue remains at the previous level until the last ACK in the cycle induce two release of packets which makes queue rise one more packet. By the analysis of 2.1, we know that if the condition (10) is satisfied, the buffer occupancy at the beginning of congestion avoidance phase is zero. Buffer size of each window epoch can only reach to one packet in congestion avoidance phase if the number of packets transmitted in the path during a window epoch is smaller than Bandwidth-Delay Product (BDP). That is, when the next cycle begins, data of the current cycle are all contained in the link, no additional buffer space is required to accommodate data in the current cycle. And there is no queuing delay effect on packet Round-Trip Delay, so we obtain the condition of the same duration epoch of each window cycle in congestion avoidance phase as follow,

$$2w_t\le uT\ . \tag{12}$$

Since the total number of packets (n) transmitted in the congestion avoidance phase is given by,

$$n=\sum_{i=1}^{w_0}(w_t+i)\,. \tag{13}$$

With the duration of congestion avoidance phase of w_TT (w_T cycles each lasts T time), the average throughput of the whole phase including the Slow-Start is then expressed as,

$$\overline{\lambda}=\frac{w_t+n}{T(\log_2 w_t+1)+Tw_t}\,. \tag{14}$$

Combining (13) and (14), we see that the average throughput is mainly determined by w_t and T. The larger w_t and the smaller T obtain higher throughput performance. Since the buffer occupancy in this phase can only reach one packet, the maximum requirement for zero loss of TCP is conditioned by Slow-Start phase which is expressed as Eq.(11).

3 Analysis Model for Token Based F-TCP in One Connection Case

3.1 Slow-Start phase

Token number is updated when a data packet arrives or leaves and if it is larger than Token Thresh. It is decreased by 1. Assuming that it takes no time to calculate the token number, since the ACKs have the same transmission rate as data in the forward direction, the Token number calculated immediately returns to the spare buffer size though each time it was just reduced by the backward served ACK. So, when *N(t)>Th*, the number of Tokens is computed based on the spare buffer size Like TCP, we also consider each mini-cycle of Token F-TCP.

$$N(t)=B-ut\,. \tag{15}$$

It takes time t_1 before *N(t)<=Th*. In Slow-Start phase, one ACK causes two releases of packets into buffer, then a packet leaves after the two packets have arrived for the arrival rate is very quick. So, when $N(t)$ reaches Th+1, the buffer size raises two packets, which then causes Token number $N(t)$ reduce to Th–1. If we assume data packets arrive and leave buffer at the same time, then the buffer only increases one packet and Token number reduces to *Th*, in this paper, we consider the former assumption, so obtain that

$$t_1=\frac{B-Th+1}{u} \tag{16}$$

and from (6) and (7),

$$W(iT+t_1)=2^{i-1}+B-Th+1, \tag{17}$$

$$q_i(t_1)=B-Th+1\,. \tag{18}$$

Then a packet leaves from the forward queue after the release of two packets caused by an ACK, the Token number at this time is then equal to *Th* which prevents the backward ACK from being sent out. Since no packet is available at the buffer input port, then the forward queue empties out one packet at rate of *u*. Two packets leave out from the queue, the number of Tokens and forward queue length are recalculated as

$$q_i(t_1+1/u)=B-Th-1\,, \tag{19}$$

$$N(t_1+1/u)=Th+1>Th\,. \tag{20}$$

The window size then increases until the ACK after delay of 1/u released and the forward queue occupancy returned to (18). Token number recalculated at this time returns to *Th*+1 which means the next arrived ACK will have to be delayed. The above procedure (18) (19) (20) is repeated after time of t_1 and Token number oscillates between *Th*-1 and *Th*+1, the buffer size varies around *B-Th*-1 and *B-Th*+1 which guarantees that no buffer overflow occurs with this mechanism. Since each ACK after time t_1 is delayed by 1/*u*, the backward queue is built up and the time duration t_2 in this period is

$$t_2=\frac{2^i-(2^{i-1}+B-Th+1)}{u/2}\,. \tag{21}$$

At the end of t_2, the buffers size returns back to *B-Th*+1 for the release of the last ACK in this cycle. Then the buffer empties out at rate of 1/*u* since there is no data available in the current window. The time given during this time is

$$t_3=\frac{B-Th+1}{u}\,. \tag{22}$$

The total time t_q for the buffer containing data is

$$t_q=t_1+t_2+t_3=\frac{2^i}{u}\,. \tag{23}$$

And the time for the packet in current window to be sent out is

$$t_w=\frac{2^i-(B-Th+1)}{u}\,. \tag{24}$$

From (23) and (24), although the time for sending out current window of data is delayed, it is smaller than the duration when buffer contains data which is the same time for normal TCP. So, if the condition of (10) is satisfied, no queuing delay in the forward and backward path will affect RTT in Slow-Start pahse. For backward ACK buffer, since the time needed to send all the packets in a window is still smaller than the round-trip time for Token F-TCP, when the ACK of the first packet of current window returns, no ACKs of previous cycle which used to initiate the sending of current window remain in the backward queue. So, it passes the backward buffer without any queuing delay to invoke the next window epoch.

3.2 Congestion avoidance phase

Since there are no long-lived packets in the queue at the beginning of congestion avoidance phase for Token F-TCP, and if it also satisfies (12), buffer size of each cycle can reach only one packet. So, no ACK holding action is required in congestion avoidance phase for Token F-TCP and it has the same queue dynamics as normal TCP. Concluding from above analysis, since Token F-TCP has the same Round-Trip time as TCP in each window epoch, the average throughput during the whole phase can also be given by (14).

4 Analysis and Numerical Results

4.1 Single connection

Figure 2 illustrates the network topology and link parameters. The one way data traffic is from server to client. The reverse one is for ACKs. TCP Maximum Segment Size is set to 536 (bytes), and it has the maximum window size of 65 536 (bytes), The window SSTHRESH is set as half of 65 535. The buffer capacity of gt1 for Token is 40,000 (bits). Token Threshold is 3 packets. Only one connection is shown in Fig.2. The simulation tool used is OPNET 6.0.L. To differentiate buffer dynamics of TCP and Token F-TCP in a figure, we use discrete line for Token F-TCP, and linear line for TCP.

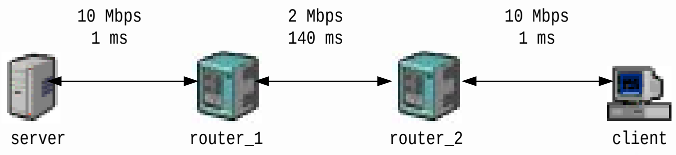

Fig.2 Simulation model for one connection case

As have analyzed, if the configuration of simulation meets the condition of (12), that is, the network Bandwidth-Delay Product is larger than the maximum window size, the time for packet stay in the queue is smaller than the RTT (Fig.3), congestion window has the same RTT of each mini-cycle for both TCP and Token F-TCP. Simulation results in Fig 4 illustrate this exactly. With TCP SSTHRESH of 32 768, it needs 7 cycles (the first cycle of 1 packet makes no built up of buffer size). Figure 4 shows that the buffer occupancy of TCP is equal to half the maximum window size of each cycle (the buffer size of first cycle is 0). For Token Based F-TCP, ACK delay starts at the fifth cycle when the buffer size is larger than *B-Th*+1 (approximately 6 packets). The buffer size is then controlled below 27 648 bits (Fig.3) by Token F-TCP mechanism that matches the analysis results from (18) exactly. The window size after forth cycle though delays its sending rate, it does not affect the duration of cycle and gains the same sequence number at each end of window epoch (Fig.5).

Since the congestion avoidance phase only occupies one packet of buffer size, no Token Delay algorithm is initiated, the buffer size, congestion window size and sent sequence number of TCP are overlapped with Token F-TCP completely (Figs.3~5) after the seventh cycle. The backward ACK buffer which is accumulated in slow start phase because of Token ACKs' delay control is elapsed to zero in congestion avoidance phase for the same reason (Fig.6).

The calculated average throughput for both TCP and Token F-TCP is 276.877 packets/sec by Eq.(14), through measuring from Fig.7, we obtain 276.8 which matches the analysis result very well.

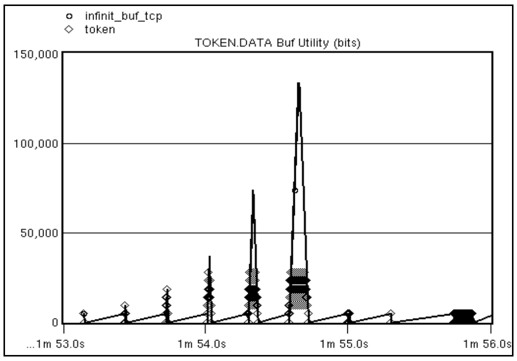

Fig.3 Forward buffer size for both TCP and Token F-TCP

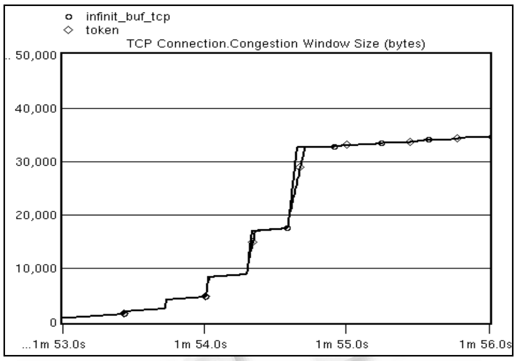

Fig. 4 Congestion window size

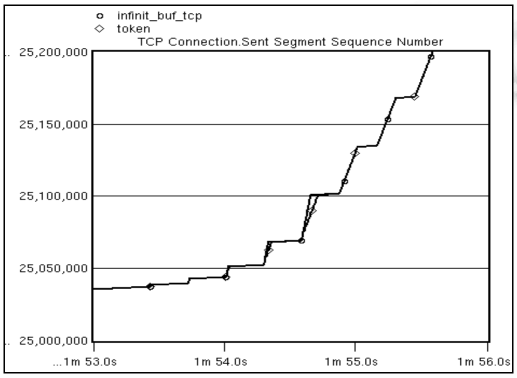

Fig. 5 Sent segment sequence number in mini-cycle

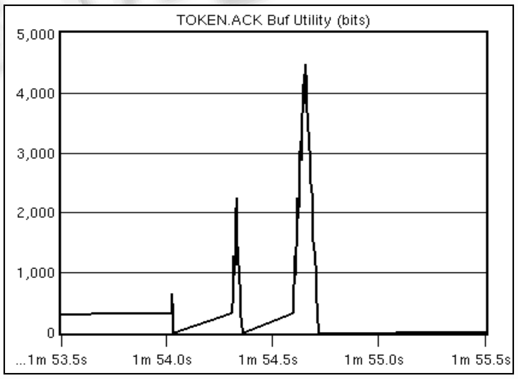

Fig. 6 Token backward ACK buffer size

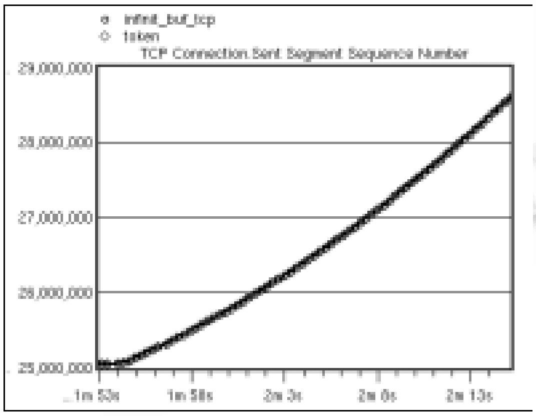

Fig. 7 Sent segment sequence number during the whole phase

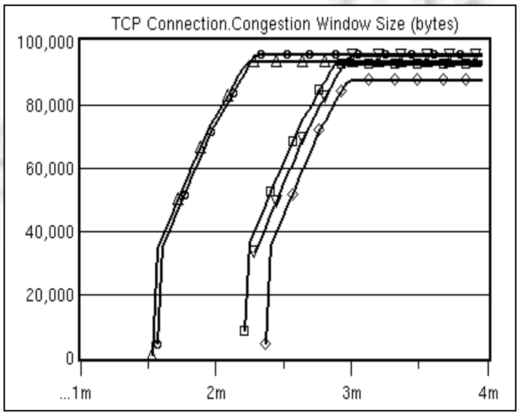

Fig. 8 Congestion window size for five connections

4.2 Multiple connections

For multiple connections, all the link rate is configured five times that of one connection to ensure that Bandwidth-Delay Product is larger than the total maximum window size of five connections. Since it is rarely possible for any two of connections to start at the same time in real networks, and because Slow-Start phase lasts such a short time that each connection has no overlap in slow start phase in our simulation. See Fig.8, connection 2 is initiated after slow start phase of connection 1 has ended though they are neighbored very closely. In congestion avoidance phase, one packet increased which is also very short in the cycle duration, so the possibility of one connection superposing another connection in the buffer is very small.

Like one connection case, buffer size is mainly dominated by Slow-Start phase of each connection (see Figs.8 and 9). For TCP, if no overlap occurs between connections, buffer has the same size as in one connection case. From Figs.8 and 9, we see that, since the maximum SSTHRESH of each connection is 32 768 bytes, the maximum buffer occupancy of buffer is 32768/2. With overlap between congestion avoidance and slow start, see Fig.9, the second initiated connection in slow start phase meets with the first initiated connection in congestion avoidance phase, and the buffer occupancy can reach one more packet than half of SSTHRESH. For Token F-TCP with multiple connections, the forward buffer occupancy can still be controlled below the B-TH+1 (27 648 bits) as one connection case (Fig.9).

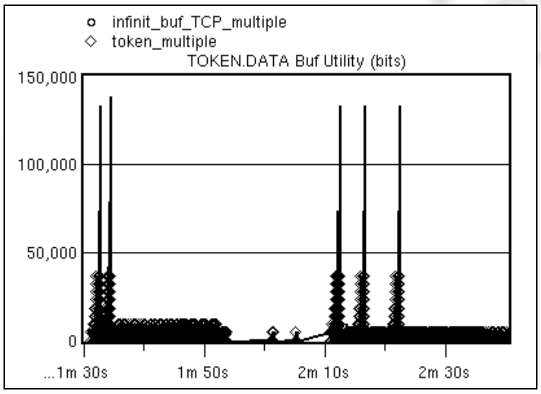

Fig. 9 Buffer occupancy for five connections

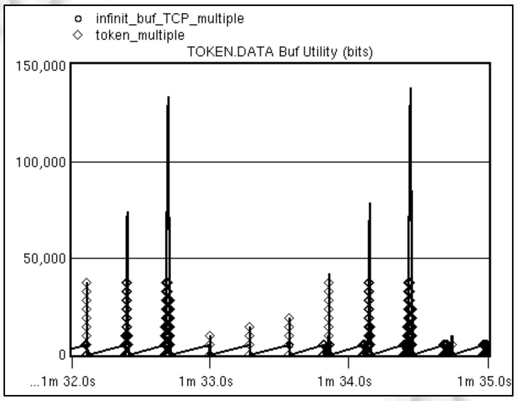

Fig. 10 Buffer dynamics for one of five connections

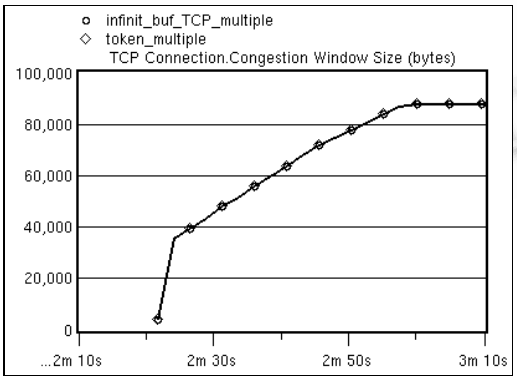

Fig. 11 Congestion window size for one connection in five connection cases

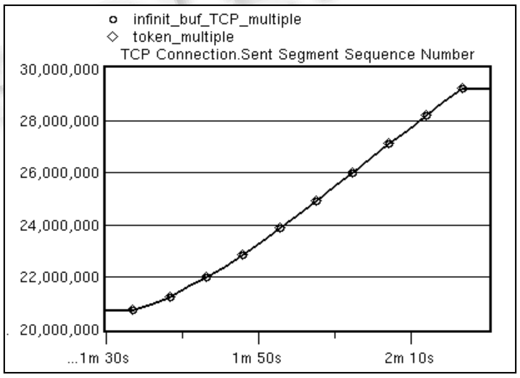

Fig. 12 Sent sequence number for one connection in five connection cases

To illustrate the analysis, we choose the first two initiated connections, and enlarge their buffer size in Fig.10. It shows that the duration of each cycle is the same as RTT like one connection, and there is no queuing delay effect even in multiple connections for both TCP and Token TCP. The congestion window size and sent sequence number of Token F-TCP matches those of TCP exactly see (Figs. 11 and 12) in each connection. Therefore, the same throughput performance for Token F-TCP is obtained as that of TCP with lower forward buffer requirement if the total maximum window size of multiple connections is smaller than Bandwidth-Delay Products for at this time all data from multiple connections can be contained in the link, the buffer only occupies packets temporally at each cycle.

5 Conclusions

In WANs such as Satellite networks where the maximum window size cannot reach the Bandwidth-Delay Products by error rate links or by limit of large window size option in TCP end-systems, buffer requirement for zero loss of TCP is half of the window SSTHRESH. With Token F-TCP, it is robust to quickly relieve the congestion and fully avoid packet loss through reducing buffer occupancy. It is proved in such condition that Token Based F-TCP obtains averagely the same throughput performance as TCP with infinite buffer size with no delay effect on system. Approximately the same results are obtained in multiple connection case if the total maximum window size of multiple connections is smaller than the long haul Bandwidth-Delay Products. Next, two way traffic and ACK buffer overflow are also the focus of our future work.

References:

[1] Ma, J. Interworking between TCP and ATM flow controls. ATM Forum/97-0960, 1997.

[2] Zhang, P., Ma, J. Token-Based Fast-TCP. Patent Application NC, 1999.

[3] Lakshman, T.V., *et al*. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. IEEE/ACM Transactions on Networking, 1997,5(3):336~350.

[4] Lakshman, T.V., *et al*. Window-Based error recovery and flow control with a slow acknowledgment channel: a study of TCP/IP performance. In: Proceedings of the IEEE INFOCOM'97. April 1997.

[5] Narvaez, P., Si, K.Y. An acknowledgement bucket scheme for regulating TCP flow over ATM. In: Proceedings of the IEEE Globecom'97. 1997.

[6] Stevens, W.R. TCP/IP Illustrated, Volume 1: The Protocols. Reading, MA: Addison-Wesley, 1994.

[7] Jacobson, V. Congestion avoidance and control. In: Proceedings of the ACM SIGCOMM'88. 1988. 314~329.

[8] Ma, J. Performance enhancement by controlling ACKs in TCP/IP over satellite network. In: Proceedings of the Inet'98. 1998.

窗口限制系统中基于令牌的 **Fast-TCP** 的延时性能分析

彭 绯, 程时瑞

(北京邮电大学 交换技术与通讯网国家重点实验室,北京 100876)

摘要: 给出了在高带宽时延乘积网络中无拥塞丢包的 TCP 缓冲区的占用量的微分方程,然后针对广域网中窗口大小小于网络带宽时延乘积和无线网络中,由于误码丢包而使窗口尺寸无法达到链路带宽时延乘积(如卫星网)的情况,分析了一种基于令牌的 Fast-TCP 机制,得出基于令牌的 ACK 延时机制,不仅能够大大减少缓冲区的占用量结论,而且可获得与 TCP 在无限缓冲区下的同样的吞吐量.还把这个结论推广到多连接的情况,在以前论文中没有涉及.

关键词: TCP;ACK;带宽-时延乘积;微分方程

中图法分类号: TP393 文献标识码: A