Regular Paper

TCP Gateway for Satellite-based Internet Service Considering Accommodation of Multiple Customers

Teruyuki Hasegawa,† Yutaka Miyake† and Toru Hasegawa†

Satellite-based Internet is one of attractive solutions because it can be deployed even in inconvenient locations. However, it is a serious problem that TCP throughput can be degraded by the large propagation delay in the satellite link. We previously proposed to introduce an intermediate gateway only in the carrier side premises to accelerate TCP throughput and confirmed its effectiveness in case of no network congestion. On the other hand, in order to accommodate many customers, it is also required to avoid network congestion caused by the extremely asymmetry of the satellite access link. In this paper we describe the new TCP gateway taking account of multiple customers accommodated by a shared satellite access link. We also confirmed that the new gateway can accelerate TCP throughput avoiding network congestion, and that more than 8 times faster throughput is achieved comparing to the case without the gateway.

1. Introduction

Recently, a demand on broadband Internet access media has been explosively increasing to obtain comfortable Internet access environment. Various types of access technologies (e.g., ADSL, CATV, etc.) have been developed one after another to meet such a demand. In these technologies, satellite-based Internet access is an attractive solution of providing such facility because it can be deployed even in the locations where the other access media are not used from the view point of costs. On the other hand, most Internet applications such as web access use TCP [1] as a transport protocol for reliable data transfer. TCP performs window-based flow control to avoid buffer overflow on the receiver side, and the maximum TCP window size is limited to 64 KB (Kilo-Bytes) originally. Moreover, many TCP connections use smaller default window size whose value is 8 KB through 24 KB. The satellite-based Internet access thus has a problem. The propagation delay between the sender and the receiver is so large, typically 300 msec in each direction, that the throughput of these applications may be degraded. For example, if the maximum window size is 16 KB and a satellite link is used in both directions, where the RTT (Round Trip Time) between the sender and the receiver is 600 msec, the theoretical throughput is limited to about 200 kbps no matter how large the bandwidth is.

For improving TCP throughput over satellite links, various TCP enhancements have been proposed [2]. In these enhancements, the window scale option [3] which enlarges the maximum window size up to 1 GB is considered to be effective and has been adopted by several operating systems recently. However, it is not expected that all hosts support this option. Even if supported, it is not easy to properly set up socket buffer size, which corresponds to the TCP window size, in both the sender and receiver terminals.

As the other approaches, there are some proposals to introduce a pair of gateways at both ends of a long distance segment (e.g., satellite link) [4)~7)]. However, from the view point of initial and maintenance costs, these approaches have a problem for satellite-based Internet access that every customer should install an additional gateway on its premises as well as on a carrier-side satellite earth station.

In order to solve this problem, we have proposed another gateway-based approach where only one gateway needs to be introduced on the satellite earth station side [8),9)]. This gateway, which we call *TCP gateway*, prefetches TCP data segments (DTs) from the sender according to the native TCP. Then, the gateway speculatively sends ahead DTs directly to the receiver beyond the advertised receive window. The receiver thus can receive DTs continuously as if the sender were located closely. We have also evaluated our first design of TCP gateway and confirmed its effectiveness. This design is optimized for the environment such that network congestion does not occur in both directions

† KDDI R&D Laboratories, Inc.

of the access link. On the other hand, many satellite-based Internet services adopt an access method such that multiple customers share a downstream satellite link. The bandwidth of the downstream link can be much larger than that of the upstream link corresponding to the number of customers. However, such extremely asymmetrical environment may cause network congestion [10]. In order to accommodate many customers, it is required to accelerate TCP throughput avoiding occurrence of network congestion.

This paper describes design and implementation of the new TCP gateway taking account of multiple customers accommodated by a shared satellite access link. The main features of our new gateway are as follows.

- For preventing network congestion and buffer overrun in the receiver, a congestion avoidance mechanism based on TCP-Vegas [11] is introduced. This includes some optimization for our *send ahead* mechanism.
- The gateway provides a set of functions to detect a situation where the advertised receive window is continuously closed (i.e., zero window), and to suspend *sending ahead* DTs during this situation in order not to cause unnecessary retransmissions.

The rest of this paper is organized as follows. Section 2 surveys our TCP gateway approach proposed in Refs. 8) and 9). Section 3 describes the details of our new design of TCP gateway. Section 4 shows some communication examples in order to verify the behavior of our new implementation. Section 5 gives conclusions of this paper.

2. TCP Gateway Overviews

2.1 Network Configuration

Figure 1 shows a network configuration of satellite-based Internet access introducing a TCP gateway. The TCP gateway is located close to a satellite access router in the earth station side, and accommodates all the traffic outgoing to and incoming from customers via this router. In this configuration, a client at the customer premises receives data from the server in the Internet via a satellite link. As for the reverse (i.e., upstream) direction, traditional access media such as PSTN or IDSN may be used instead of a satellite link. Even in case of using a satellite link for the upstream direction, the bandwidth is smaller than the downstream satellite link due to the limitation of antenna size and transmission power of the customer premises. In addition, the main purpose of Internet access is to download information from the Internet. Therefore, high TCP throughput is expected for data transmission from Internet servers to customer-side clients.

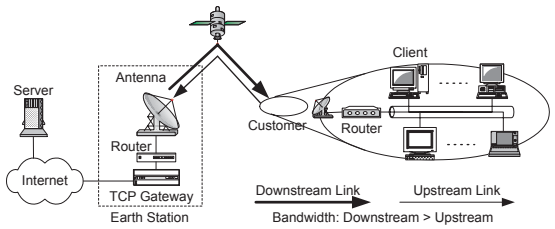

Fig. 1 Network configuration using TCP gateway.

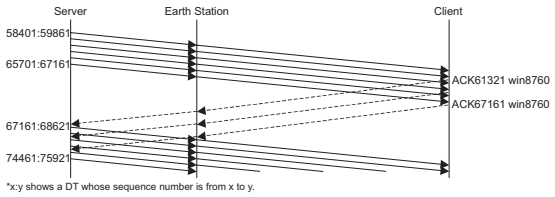

Fig. 2 TCP communication example.

2.2 Communication Sequence

Figure 2 shows a typical TCP communication sequence without TCP gateway. Due to large propagation delay and insufficient TCP window size, the server (i.e., sender) cannot help suspending DT transmission until an acknowledgement segment (ACK) including window update returns from the client (i.e., receiver). As a result, TCP throughput is saturated even if the satellite link bandwidth is large enough.

In order to accelerate the throughput of TCP data transmission from Internet servers to customer clients, a TCP gateway is introduced next to the router in the earth station as shown in Fig. 1. **Figure 3** illustrates how the gateway boosts the data transmission. The communication sequence is specified as follows.

(1) Originally, flow control and retransmission functions are provided by TCP on an end-to-end basis. On the contrary, the gateway splits these flow control and retransmission loops into the following 2 sections: between the server and the gateway, and between the gateway and the client.
(2) Instead of the client, the gateway returns ACKs to the server according to

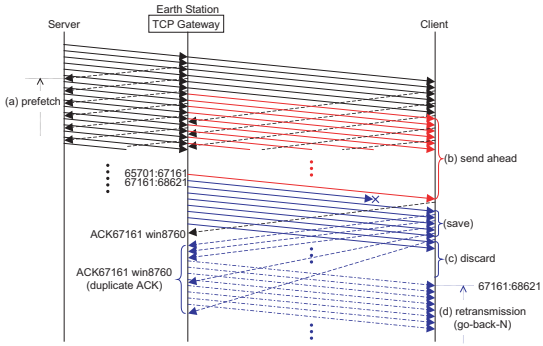

Fig. 3 Communication example using TCP gateway.

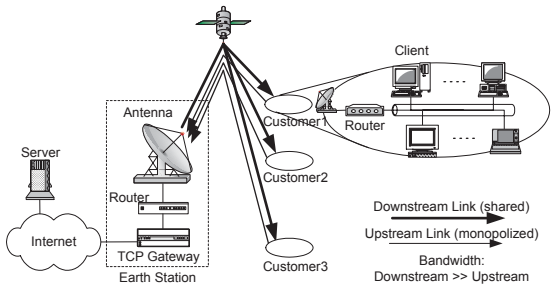

Fig. 4 Accommodation of multiple customers.

native TCP so that the server can update receive window as quick as possible (Fig. 3 (a)). As a result, the gateway can prefetch DTs from the server.

(3) The gateway speculatively sends ahead DTs to the client beyond the advertised receive window. Although this violates the TCP flow control mechanism, it is expected that the client have replied ACKs before receiving these *sent ahead* DTs (Fig. 3 (b)). Even if some DTs are lost, the client will discard the following DTs received but out of window along the TCP specification (Fig. 3 (c)).

(4) The gateway finds the DT loss by timeout or receiving 3 duplicate ACKs like native TCP. Since most DTs following the lost one are discarded due to out of window, the gateway retransmits all the DTs from the lost one on a go-back-N basis (Fig. 3 (d)). On retransmission, the gateway counts the number of DTs within the window in order to avoid unnecessary retransmission.

It should be noted that all the end hosts need not be aware of the presence of the gateway because the gateway does not change IP addresses and TCP port numbers in each TCP segment.

3. New TCP Gateway Design

3.1 Limitation of the Previous Design

Since our previous gateway design assumes that no network congestion occurs in both directions of the access link, TCP slow start and congestion avoidance functions [13] are omitted. The gateway simply transmits or retransmits DTs at a fixed rate whose value is decided based on either bandwidth of downstream or upstream access link.

On the other hand, many satellite-based Internet services adopt an access method such that a downstream satellite link is shared by multiple customers as shown in **Fig. 4**. Here, the more customers are accommodated, the larger bandwidth is required for the downstream satellite link. This increases asymmetry of link bandwidth between the downstream and the upstream. Such environment may cause serious network congestion in the upstream link by ACKs when most TCP traffic rushes into a certain customer.

The larger bandwidth also increases the possibility of segment loss caused by receive buffer overrun in the client. As described in Section 2.2, the penalty of segment loss is too expensive in our approach.

In addition, receive buffer overrun also occurs when the client application stops the DT transmission from the server based on the TCP flow control. For example, a file download through a web browser uses this mechanism as shown in **Fig. 5**. Although the client TCP returns several ACKs with zero receive window (we call it *zero window ACK* or *ZWA* for short) until the user decides the file name for output, the gateway sends ahead the following DTs only to be discarded in the client. Furthermore, the gateway retransmits these DTs even if the receive window is still closed.

3.2 Design Principles

Considering the above limitations, we adopt the following design principles for our new TCP gateway [12].

(1) In order to share the downstream satellite link fairly and effectively among a lot of customers, we should introduce some congestion avoidance mechanism into the new TCP gateway. In addition, the mechanism should also conform to our go-back-N based retransmission policy.

(2) Since the segment loss penalty is serious in the gateway, it is expected that

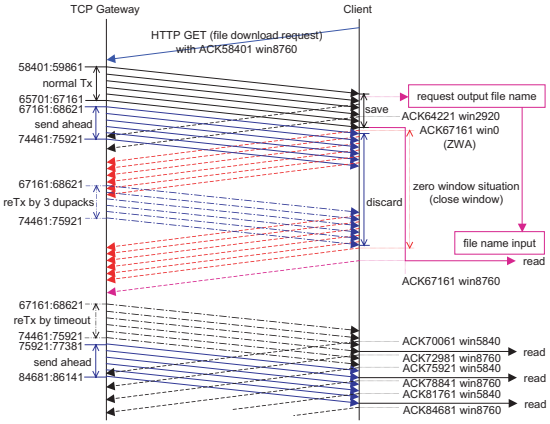

Fig. 5 File download using web browser.

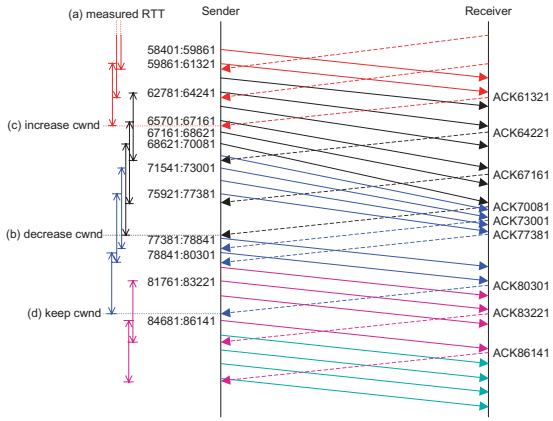

Fig. 6 Congestion avoidance mechanism in TCP-Vegas.

the congestion avoidance mechanism can work before segment loss. In other words, different from most of TCP implementations such as TCP-Reno, the gateway should not detect congestion only by segment loss.

(3) Unnecessary retransmission caused by zero window ACKs should be prevented. Thus, it is required for some mechanism which suspends and resumes retransmission by detecting persistent receive window closing and re-opening situations in the client, respectively.

3.3 Congestion Avoidance Mechanism

3.3.1 TCP-Vegas

According to the above principles, we have designed the congestion avoidance mechanism based on TCP-Vegas [11]. In TCP-Vegas, congestion avoidance is executed not only by segment loss but also by RTT fluctuation. **Figure 6** and the following description show how the RTT-based congestion avoidance mechanism works in TCP-Vegas.

(1) The mechanism also uses the congestion window (*cwnd*) arranged for the procedures specified in Ref. 13). The sender cannot transmit DTs beyond the *cwnd* as well as the advertised receive window.

(2) The sender records the sending time of every DT and measures the RTT value on receiving the corresponding ACK (Fig. 6 (a)). The minimum RTT value is recorded as RTT_{base}.

(3) The sender calculates the average value of RTT (RTT_{ave}) when it receives an ACK whose acknowledgement number is ahead of all the outstanding DTs sending before the previous calculation (Fig. 6 (b)-(d)). That is, RTT_{ave} is calculated once per RTT.

(4) Simultaneously, the sender calculates the expected rate ($rate_{exp}$) and the actual rate ($rate_{act}$)using the amount of DTs transmitted after the previous RTT_{ave} calculation.

$$rate_{act} = (amount\ of\ DTs)/RTT_{ave}$$
$$rate_{exp} = (amount\ of\ DTs)/RTT_{base}$$

(5) The *cwnd* is adjusted according to the following 3 cases.

$$rate_{exp} - rate_{act} > \beta \quad (1)$$
$$rate_{exp} - rate_{act} \leq \alpha \quad (2)$$
$$\alpha < rate_{exp} - rate_{act} \leq \beta \quad (3)$$

Here, α and β are fixed threshold values for detecting network congestion or its termination. In case of (1), the *cwnd* is considered to be so excessive that DTs are buffering on the network. The sender slows down the transmission rate by decreasing the *cwnd* by 1 (Fig. 6 (b)). On the contrary, in case of (2), the sender speeds up the rate by increasing the *cwnd* by *1/cwnd* on every ACK receiving (Fig. 6 (c)). In case of (3), the sender does not change the *cwnd* (Fig. 6 (d)).

3.3.2 Customization for TCP Gateway

We have customized the above mechanism for our new TCP gateway as follows.

(1) In a statellite link, the *cwnd* is expected to increase as quick as possible at first. We adopt the original slow start mechanism specified in Ref. 13) because TCP-Vegas' slow start mechanism is quite conservative [11].

(2) When a TCP connection has been established, the gateway sets the initial value ($cwnd_{init}$) to the *cwnd*. In default, the $cwnd_{init}$ is 4 for rapid increase of the *cwnd*. The gateway can send DTs up to the send window (*sndwin*) given by the following equation.

$$sndwin = min(awin+offset, cwnd)$$

Here, the *awin* and the *offset* show the advertised receive window and the amount of DTs which the gateway can send ahead, respectively.

(3) As for the value of *ssthresh* which indicates the boundary between the slow start and the congestion avoidance, the gateway sets the following value.

$$ssthresh = min(c \times BDP_{downstream}/cnum, offset)$$

Here, the *cnum* means the number of TCP connections which the gateway is handling. The $BDP_{downstream}$ indicates the bandwidth delay product for the downstream link. The value c is a constant to decide how far the downstream link is overbooked. Because the gateway treats all the TCP connections, it is possible to know how many connections exist and to set an appropriate target value of the *ssthresh* at the slow start phase.

(4) During the slow start, the value of *cwnd* is increased by 1 on every ACK arrival. This exponentially increases the *cwnd*. If either of the following conditions is satisfied, The gateway moves into the congestion avoidance phase.

$$cwnd \geq ssthresh$$
$$rate_{exp} - rate_{act} \geq \alpha$$

(5) During the congestion avoidance, the gateway follows the procedure described in Section 3.3.1 for linear increase or decrease of the *cwnd*.

(6) On detecting segment loss, the gateway changes the *ssthresh* and the *cwnd* as follows.

$$ssthresh = cwnd/2$$
$$cwnd = cwnd_{min}\ (e.g., 2)$$

Here, the minimum value of *cwnd* ($cwnd_{min}$) is a pre-defined constant value.

(7) If the loss is detected by duplicate ACKs, the gateway increases the *cwnd* by *1/incr* on receiving a further duplicate ACK.

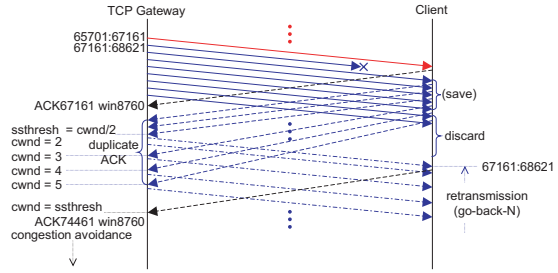

Fig. 7 Retransmission sequence.

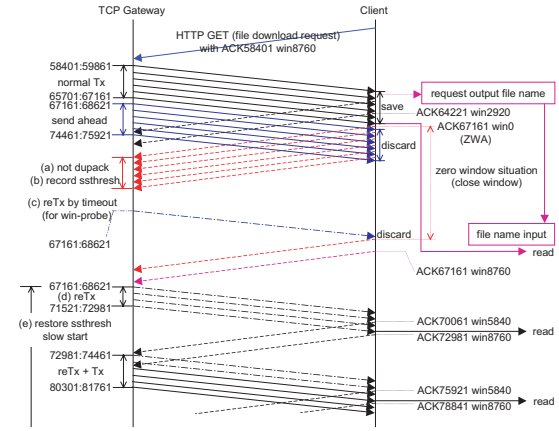

Fig. 8 ZWA manipulation in TCP gateway.

Here, the *incr* is a pre-defined constant value which decides the retransmission rate. For example, **Fig. 7** shows the retransmission sequence when *incr* is 1, where one DT is retransmitted corresponding to one duplicate ACK. Since the interval of duplicate ACKs is related to the acceptable DT rate at the receiver, the sender can retransmit DTs in accordance with the receiver's capability. After receiving an ACK for the first retransmission, the gateway moves back to the congestion avoidance phase by setting the *ssthresh* to the *cwnd*.

(8) In case of the loss detection based on a timeout, the gateway does not change *cwnd* from $cwnd_{min}$ until the first retransmission is acknowledged. After the acknowledgement, the slow start is applied.

3.4 Suspending Transmission on Zero Window Situation

Figure 8 shows an communication example when a persistent zero window situation occurs in the receiver. In order to avoid unnecessary (re) transmission, the gateway manipulates a ZWA (ACK whose window size field is 0) as

follows.

(1) The receiver informs no vacancy of the receive buffer with a ZWA. The gateway does not count the ZWA as a duplicate ACK. This suspends DT retransmission during the zero window situation (Fig. 8 (a)).

(2) Instead, ZWAs received continuously are counted up using another counter. If this counter exceeds the threshold *zwacnt* (e.g., 10 ZWAs), the gateway does not regard the following ZWAs as the trigger of *cwnd* increase. This suspends transmission of new DTs during the zero window situation.

(3) The gateway stores the *ssthresh* into the $ssthresh_{old}$ and records the ZWA arrival time (Fig. 8 (b)).

(4) If the receive window is closed on the retransmission timeout, the gateway sets the *cwnd* to 1 for probing the receive window update (Fig. 8 (c)). Different from the 6th item of Section 3.3.2, the *ssthresh* does not change.

(5) On recovery from the zero window situation, the gateway measures duration of the situation. If this duration is more than the threshold *zwadur* (e.g., 1 sec), the gateway considers that the receiver has quited the persistent zero window situation and starts go-back-N retransmission for sent ahead DTs (Fig. 8 (d)). Otherwise, the gateway resets the arrival time of the ZWA regarding the situation as temporal.

(6) The retransmission is performed with slow start using the following parameters in order to recover transmission rate quickly (Fig. 8 (e)).

$$ssthresh = ssthresh_{old}$$
$$cwnd = cwnd_{init}$$

4. Performance Verification

4.1 Configuration and Results

We have implemented the new TCP gateway based on our first implementation. The gateway is a software-based implementation working as a kernel part of Solaris 2.6 OS.

In order to verify the behavior of our congestion avoidance mechanism, we confirmed this implementation using a customer with 4 windows 98 clients accommodated with an asymmetrical access link emulated by ATM as shown in **Fig. 9**. The downstream link bandwidth is 10 Mbps, and the upstream one is 150 kbps. By use of a data channel simulator (ADTECH SX-14), a constant propagation delay (700 msec RTT) is inserted. Thus the value $BDP_{downstream}$ becomes 875 KB. As for the parameters in our congestion avoidance mechanism, we used the following values:

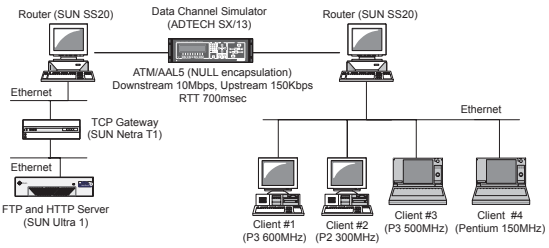

Fig. 9 Network configuration for verification.

- $offset = 128$ KB for sending DTs ahead beyond the advertised receive window,
- $c = 1$ for sharing downstream link bandwidth conservatively,
- $cwnd_{init} = 4$ and $cwnd_{min} = 2$, and
- $\alpha = 17$ kbps and $\beta = 50$ kbps by which the gateway tries to keep the number of DT segments queued in the network between 1 and 3.

Figure 10 shows the transition of the *cwnd* and the transmission rate applied to one of 4 clients when these clients simultaneously download a 4 MB file using FTP. The *cwnd* values applied to all 4 clients finally approach 80 KB. The total TCP throughput in 4 clients is accelerated to 370 KB/sec by use of the gateway. On the other hand, the total throughput without the gateway is resulted in 46 KB/sec.

In addition, we also verified our new algorithm for a persistent zero window situation using 1 client (client #1) in Fig. 9 with the following parameter values:

- $zwacnt = 10$ ZWAs, and
- $zwadur = 1$ sec where we assume that it takes several or more seconds to input the download file name.

In case of downloading a 4 MB file using a web browser (IE 5.0), the TCP throughput observed in this client is resulted in 48 KB/sec when the gateway does not use this algorithm. On the contrary, 117 KB/sec throughput is achieved when this algorithm is enabled.

4.2 Discussions

(1) The results of our first verification prove that the new gateway can control the TCP traffic volume properly using our modified TCP-Vegas based congestion

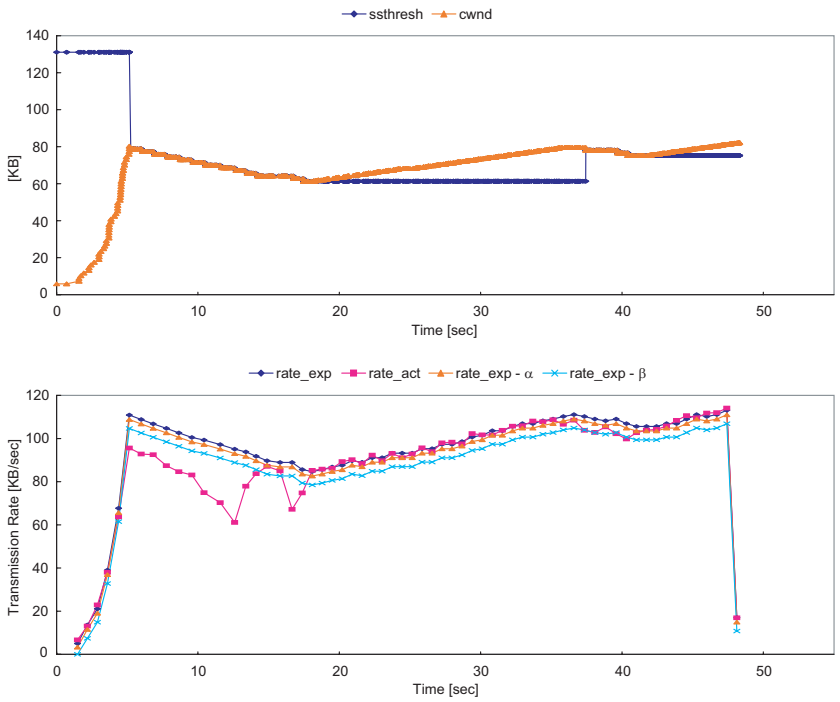

Fig. 10 Transition of cwnd and transmission rate.

avoidance mechanism, as well as can accelerate the total TCP throughput from 46 KB/sec to 370 KB/sec. However, it should be noted that the new gateway cannot attain so high throughput as our previous implementation. According to the the results described in 10), 570 KB/sec throughput is expected using the previous gateway. On the other hand, the previous gateway requires an average of 150 msec additional delay due to upstream link congestion, while the new gateway requires 50 msec. Thus there is some trade-off between the throughput and the additional delay. However we believe that our congestion avoidance mechanism is inevitable for the environment where multiple customers share a downstream link because the gateway should adapt the traffic demand on each customer dynamically.

(2) The results of our second verification show that our new ZWA manipulation algorithm is very effective for improving TCP throughput in web-based file downloading with file name input. According to the results in the first verification, the throughput without the gateway in the second verification is estimated at 12 KB/sec. Therefore, the gateway with this algorithm can give about 10 times faster throughput. However it should be noted that this algorithm strongly depends on its parameter values, especially on *zwadur*. That is, zero window situation whose duration is shorter than *zwadur* cannot be detected. On the contrary, small *zwadur* (e.g., less than 100 msec) may cause mis-detection of such a situation even when the TCP receive buffer in the client is temporally full. Therefore the value of *zwadur* should be selected carefully considering the application types. As for the web-based file downloading, our results indicate that $zwadur = 1$ sec has no problem for detecting a persistent zero window situation.

(3) It is considered that the TCP-Vegas based mechanism cannot work properly in the following cases because this controls the volume of TCP traffic based on measured RTTs.

(a) TCP-Reno based traffic, which continues to increase traffic volume until segment loss occurs, is multiplexed in a bottleneck link.

(b) RTTs are changed by some fac-

tors other than network congestion. For example, some data links provide unneglectable delay variation due to their resource allocation mechanisms.

On the other hand, we assume the network configuration as follows in this study. Therefore we consider that the TCP-Vegas can be applied in our gateway system.

(a) As described in Section 2.1, the TCP gateway accommodates all the traffic outgoing to and incoming from customers. This means that all the TCP traffic is controlled by TCP-Vegas in satellite links.

(b) The link capacities of the downstream and upstream satellite links are statically pre-assigned and propagation delays in these links are constant.

5. Conclusions

This paper describes our new TCP gateway design considering that multiple customers accommodated by a shared satellite access link, where network congestion can cause due to extreme asymmetry of access link. We have introduced a congestion avoidance mechanism based on TCP-Vegas with some customization mainly with regard to our go-back-N retransmission mechanism. In addition, we have also developed and verified the new algorithm for a persistent zero window situation in order not to cause unnecessary retransmissions and throughput degradation. By use of these mechanisms, the new gateway can control the TCP traffic volume properly before (or with the minimum) segment losses in many cases. TCP throughput is accelerated more than 8 times in our verification.

Acknowledgments The authors wish to thank Mr. T. Asami, President & CEO of KDDI R & D Laboratories Inc., for the continuous encouragement of this study.

References

1) Postel, J.: Transmission Control Protocol, RFC793 (Sept. 1981).
2) Allman, M., Glover, D. and Sanchez, L.: Enhancing TCP over satellite channels using standard mechanisms, RFC2488 (Jan. 1999).
3) Jacobson, V., Braden, R. and Borman, D.: TCP extensions for high performance, RFC1323 (May 1992).
4) Henderson, T. and Katz, R.: Transport protocols for Internet-compatible satellite networks, *IEEE Journal on Selected Areas in Communications*, Vol.17, No.2, pp.326–344 (Feb. 1999).
5) Hasegawa, T., Hasegawa, T., Kato, T. and Suzuki, K.: Implementation and performance evaluation of TCP gateway for LAN interconnection through wide area ATM network, *IEICE Trans.*, Vol.J79–B–I, No.5, pp.2330–2341 (May 1996) (in Japanese).
6) Kato, T., Hokamura, K., Yamada, M., Hasegawa, T., Hasegawa, T. and Sawada, K.: TCP gateway improving throughput of TCP/IP over wide area ATM networks, *ISS* (*International Switching Symposium*) *'97*, Vol.1, pp.35–42 (Sept. 1997).
7) Hasegawa, T., Hasegawa, T., Kato, T., Yoshiizumi, K., Miki, T., Hokamura, K. and Sawada, K.: Protocol architecture of high speed TCP/IP service over international ATM network, *'98 ATM Workshop*, pp.159–168 (May 1998).
8) Miyake, Y., Hasegawa, T., Hasegawa, T. and Kato, T.: Acceleration of TCP throughput over satellite-based Internet access, *ISCC* (*IEEE Symposium on Computers and Communications*) *2000*, pp.245–253 (July 2000).
9) Miyake, Y., Hasegawa, T., Hasegawa, T. and Kato, T.: Proposal of TCP gateway for satellite-based Internet access, *IEICE Trans.*, Vol.J84–B, No.12, pp.2330–2341 (Dec.2001) (in Japanese).
10) Hasegawa, T., Miyake, Y., Hasegawa, T. and Nakao, K.: A study on accommodating multiple destinations in TCP gateway for satellite-based Internet, *The 62th National Convention IPSJ*, Vol.3, pp.385–386 (March 2001) (in Japanese).
11) Brakmo, L. and Peterson, L.: TCP Vegas: end to end congestion avoidance on a global Internet, *IEEE Journal on Selected Areas in Communications*, Vol.13, No.8, pp.1465–1480 (Oct. 1995).
12) Hasegawa, T., Hasegawa, T., Miyake, Y. and Nakao, K.: TCP Gateway for Satellite-based Internet Service Accommodating Multiple Subscribers, *WCNC* (*Wiress Communications and Networking Conference*) *2002*, Vol.2, pp.849–854 (March 2002).
13) Stevens, W.: TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms, RFC2001 (Jan. 1997).

(Received March 26, 2002)
(Accepted October 7, 2002)

Teruyuki Hasegawa received the B.E. and M.E. degrees of electrical engineering from Kyoto University, Japan, in 1991 and 1993 respectively. Since joining KDD in 1993, he has been working in the field of high speed communication protocol and ATM. He is currently a research engineer of Network Management System Lab. in KDDI R&D Laboratories Inc. He received Best Paper Award for Young Researchers of the National Convention of IPSJ in 1996, Best Paper Award of the National Convention of IPSJ in 1999 and 2002, and Young Engineer Award of IEICE in 2000. He is a member of IEICE.

Yutaka Miyake received the B.E. and M.E. degrees of Electrical Engineering from Keio University, Japan, in 1988 and 1990, respectively. He joined KDD (now KDDI) in 1990, and has been engaged in the research on high speed communication protocol, secure communication and privacy protection system for network applications. During a year of 2000 to 2001, he was a visiting researcher at University of California, Berkeley. He is currently a senior research engineer of Network Security Lab. in KDDI R&D Laboratories Inc. He received IPSJ Convention Award in 1995. He is a member of IEICE.

Toru Hasegawa received the B.E., M.E. and Dr. Informatics degrees in information engineering from Kyoto University, Japan, in 1982, 1984 and 2000, respectively. Since joining KDD in 1984, he has been working in the field of formal description technique (FDT) of communication protocols. From 1990 to 1991, he was a visiting researcher at Columbia University. His current interests are mobile computing and high-speed protocol. He is currently the Senior Manager of Network Management Lab. in KDDI R&D Laboratories Inc. He is also a guest professor at National Institute of Informatics. He received IPSJ Convention Award in 1987. He is a member of IEICE.