Recommended Paper

Spurious Timeout Detection Algorithm for Mobile Communication with Delay Jitter

Motoharu Miyake,[†] Hiroshi Inamura[†] and Osamu Takahashi[††]

A spurious timeout (STO) leads to an unnecessary go-back-N retransmission and throughput degradation, which negatively impacts the user's TCP performance. In this paper, we propose an STO detection and congestion window control algorithm based on the first acknowledgment following an RTO monitoring event for suppressing both the unnecessary retransmission and throughput degradation. This method strongly supports the enhancement of existing mobile communications systems because it does not require additional information or receiver modification.

1. Introduction

As mobile communications are now extremely popular, the significance of the impact of a wireless network's characteristics on TCP performance has become of great importance. In order to provide reliable transfer, one of the global telecom systems for the IMT2000 3G mobile communication standard, wideband code division multiple access (WCDMA), uses radio link control (RLC), a selective repeat and sliding window auto repeat request (ARQ) 1) scheme. The ARQ mechanism in WCDMA supports a packet service with a negligibly small probability of undetected errors due to RLC frame retransmission 2). However, the delay jitter caused by error recovery can lead to an unexpected increase in round trip time (RTT). In this event, the TCP sender experiences a retransmission timeout (RTO) because it has no information about the wireless conditions. The sender then assumes that outstanding segments are lost, and the unacknowledged segment is retransmitted.

In an RTO event, the base station (BS) keeps the original and retransmitted TCP segments until the recovery of wireless link, and then sends them to the receiver. When the sender does not use a TCP timestamp option, it can not identify if the acknowledgment is in the response to the original or the retransmitted segment. This raises the retransmission ambiguity problem 3), and the sender continues to retransmit the unacknowledged segments in response to each acknowledgment. This leads to the unnecessary go-back-N retransmission, which degrades the throughput because of the congestion window size reduction. This RTO event is called spurious timeout (STO). WCDMA is likely to experience such STO problems due to delay jitter (e.g., error recovery with RLC frame retransmission or passage outside of the service area) 1). On the other hand, the delay jitter caused by a packet retransmission in the wireless LAN environment such as IEEE802.11b 4),5), leads to a limited increase even if the "Backoff Time" reaches the maximum value. Wireless LANs are not likely to experience STO problems, so that this paper makes no mention of them. In order to suppress the STO problem, several algorithms based on explicit and implicit information have been proposed 3),6)~8).

We note that due to the RLC setting of persistency in the BS, the ARQ may give up retransmission which leads to TCP segment loss. This occurs, for example, if link layer transmission does not succeed within a period of time or a number of transmissions due to an extreme poor wireless link 2). In this event, the first unacknowledged segment is discarded, and the receiver acknowledges the series of following segments as duplicate ACKs. The conventional STO detection algorithms 3),6)~8) have no means of detecting the duplicate ACKs that follow the RTO, and they revert to the conventional algorithm, even if one segment is lost while the others have reached the receiver. In

† NTT DoCoMo Inc.
†† Future University-Hakodate

The initial version of this paper was presented at the 24th conference held on March 2003, which was sponsored by SIGMBL. This paper was recommended to be submitted to IPSJ Journal by the steering member of SIGMBL.

this case, the TCP sender directly enters the conventional avoidance phase without the slow start phase after RTO. This degrades throughput because the sender waits for the first acceptable ACK following RTO and restarts segment transmission with a small congestion window size.

To avoid these significant TCP performance problems, we propose an STO detection and congestion window control algorithm that suppresses unnecessary retransmission and the throughput degradation caused by STO and duplicate ACKs following RTO arrival. It is based on implicit information collected by monitoring the first acknowledgment, such as an earlier ACK, an acknowledgment follows the delayed ACK that covers 2 full-size segments and up, and duplicate ACKs following an RTO[9]. In Allman's report, most of the acknowledgments that follow the first retransmission after RTO within the period $RTT_{min}/2$ indicate STO in wired networks[7]. In addition, the characteristic of an acknowledgment arrival time can distinguish the time between segment retransmission via the slow start algorithm and the arrival of the acknowledgment. In the case of the first acknowledgment to cover the arrival of 2 full-size segments, it means the response to original outstanding segments according to the delayed ACK[10] if the bottleneck wireless link provides sufficient bandwidth[9]. Moreover, duplicate ACKs following the RTO usually are the response to the original outstanding segments according to the expiration of ARQ retransmission.

The proposed algorithm requires the Reno algorithm; the only modifications required are to the TCP protocol stack of the sender. It does not require any additional information such as special bits in TCP/IP header fields. Accordingly, it is easy to support current receiver side equipment, such as Internet access cellular phones, PDAs, and PCs. In addition, it can make use the Selective Acknowledgement (SACK) option[11] and the duplicate selective acknowledgment (DSACK) option[12] to enhance the detection accuracy.

We implemented the proposed algorithm in the *ns*2 simulator[13] to confirm its performance improvement, and show the effect of suppressing unnecessary retransmission. Furthermore, we assessed the throughput improvement by the algorithm's congestion window size behavior. The results confirm that the proposed algorithm can avoid the throughput degradation more effectively than the conventional algorithms.

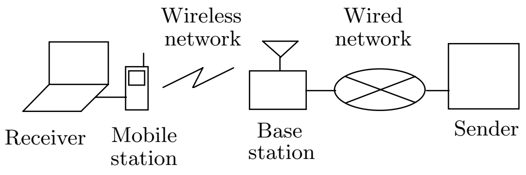

Fig. 1 System model.

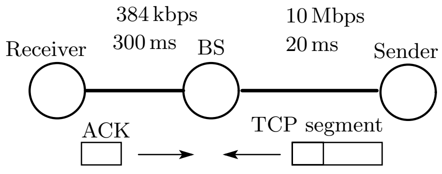

Fig. 2 Simulation model.

2. System Model and STO

Figure 1 shows the system model of mobile communication considered here. The TCP sender in the wired network communicates with the receiver in the wireless network who is using a protocol such as WCDMA. WCDMA is likely to experience STO problems caused by the delay jitter that accompanies error recovery with RLC frame retransmission or passage outside of the service area[1]. **Figure 2** shows the topology used in our experiments. The sender is connected to the BS via a 10 Mbps wired link with a RTT of 20 ms, and the receiver is connected to the BS via a 384 kbps wireless link with a RTT of 300 ms. The TCP segments are transmitted from the sender to the receiver.

The sender measures the RTT for TCP timeout and retransmission. This is needed for handling route changes and network traffic changes, however the sender can experience the STO problem easily. This is because it is hard to support unexpected RTT increments caused by the delay jitter the accompanies error recovery with RLC frame retransmission or passage outside of the service area. This means that the sender retransmits the outstanding segments even if the BS has enough queue depth and has not dropped any original outstanding segments. When if a sender without timestamp option receives an acknowledgment following an RTO, it faces the retransmission ambiguity problem[3]. It assumes that outstanding segments are lost, and retransmits unacknowl-

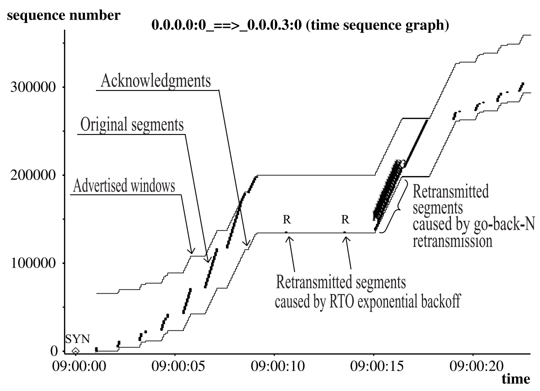

Fig. 3 An example of unnecessary retransmission and throughput degradation caused by an STO.

edged segments. Moreover, the sender continues to retransmit the unacknowledged segments in response to each of the acknowledgments which leads to unnecessary go-back-N retransmission. The STO problems during bulk data transfer in the GPRS (General Packet Radio Service), which has similar ARQ retransmission mechanism to the W-CDMA, is reported in Ref. 14).

Figure 3 shows conventional TCP sender behavior after an STO. The graph convention used in the time-sequence graph is similar to that introduced in Ref. 15). The thick and thin solid lines in Fig. 3 plot the segments, acknowledgments, and advertised window size, respectively. Symbol "R" shows the segment retransmitted by the exponential backoff algorithm and go-back-N retransmission. This trace was captured by the *ns*2 simulator on the sender side. The figure shows go-back-N retransmission after exponential backoff and the reduction in the congestion window size after the RTO. As a result, the sender performs unnecessary retransmission and the throughput degrades.

3. Related Work

In order to suppress the STO problem, several algorithms based on explicit information, such as TCP options, and implicit information have been proposed.

3.1 STO Detection Based on Explicit Information

The Eifel algorithm[3] with the TCP timestamp option can identify if the acknowledgment is in response to the original segment or the retransmitted segment. The timestamp option is standardized as RFC1323, and it is implemented in most operating systems. To use the Eifel algorithm, only the sender needs to implement it. If a sender with the Eifel algorithm detects STO, it reverts to the congestion window (`cwnd`) and the slow start threshold (`ssthresh`) to avoid unnecessary retransmission and throughput degradation. Moreover it can adjust parameters for setting the RTO, to prevent more unnecessary RTO.

STO detection based on DSACK[6] is discussed in the Internet Engineering Task Force (IETF). DSACK[12] was originally proposed by S. Floyd, et al. as an extension of the SACK option[11]. The receiver reports duplicate segment information using SACK blocks, so the sender can identify that the acknowledgment is in the response to the retransmitted segment. If a sender with the DSACK detects STO, it reverts to `cwnd` as in the Eifel algorithm.

STO detection based on DSACK takes one round trip time after the first segment of go-back-N retransmission, because STO is identified from the information of the arrival of the duplicate segment at the receiver. This means that some unnecessary retransmission is unavoidable, so the Eifel algorithm detect STO more rapidly than DSACK. However, the Eifel algorithm incurs the cost of the 12 byte timestamps. The timestamp option occupies about 1 percent of a full-size segment, but this rises to about 30 percent in an acknowledgment. It is better to use the Eifel algorithm when the unnecessary retransmission caused by STO exceeds the timestamp option's overhead.

3.2 STO Detection Based on Implicit Information

Allman, et al. reported an approach for STO detection that builds on research in Internet traffic measurements[7]. Most of the acknowledgments that follow the first retransmission after an RTO within half of the minimum RTT ($RTT_{min}/2$) are responses to the original segment[7]. His definition, which states that acknowledgment arrival within $RTT_{min}/2$ indicates the existence of STO, is implemented in FreeBSD 4.3 and later versions. If a sender with Allman's algorithm detects STO, it reverts to the original `cwnd` and `ssthresh` as in the Eifel algorithm, however, it can work well only for a period after the unacknowledged segment is retransmitted.

Sarolahti, et al. proposed the forward RTO recovery (F-RTO) algorithm[8]. It sends two new segments in response to the first acceptable ACK, and monitors the response as part of STO

Table 1 Comparison between the proposed and conventional algorithms.

Algorithm	Eifel	DSACK	Allman	F-RTO	Proposed
Detection	Timestamp	SACK block	Early arrival ACK*	2nd ACK	Early arrival ACK 2 segments ACK Duplicate ACKs
Recovery	Congestion window controls (Restore original cwnd and ssthresh)				

* An acceptable ACK following 1st retransmission after a RTO only

detection. If it receives the second acknowledgment as a series of acknowledgments, it posits an STO event and reverts to the original `cwnd` and `ssthresh` as in the Eifel algorithm. Implementing the F-RTO algorithm needs only sender side modification; no TCP options or receiver side modifications are needed. This means that it can support existing receiver side equipment without any tradeoff, even if they do not support the timestamp option, DSACK, and so on. However, a sender using the F-RTO algorithm can not work well given current network problems (i.e., a duplicate ACK, or key segments reordering, or and an acknowledgment that covers all outstanding segment arrivals[6]), and it reverts to the conventional RTO recovery.

4. Proposed STO Detection Algorithm

In order to avoid these significant TCP performance problems, we propose an STO detection and congestion window control algorithm that suppresses unnecessary retransmission and the throughput degradation caused by STO and duplicate ACKs following RTO arrival. The differences between the proposed and conventional algorithms are shown in **Table 1**. The proposed algorithm monitors three type of acknowledgments, early arrival ACK, an acknowledgment following a delayed ACK that covers 2 (and up) full-size segments, and duplicate ACKs. This allows the algorithm to avoid the overhead problem while supporting current equipment without any TCP option requirements.

4.1 STO Detection Based on Early Arriving ACK

Allman's concept of STO detection in wired networks is based on the acknowledgments that follow the first retransmission after an RTO within $RTT_{min}/2$[7]. We note, however, that the sender may be forced in calling an STO event if the BS keeps all of the outstanding segments for a long time while waiting for the recovery of the wireless links. When the link

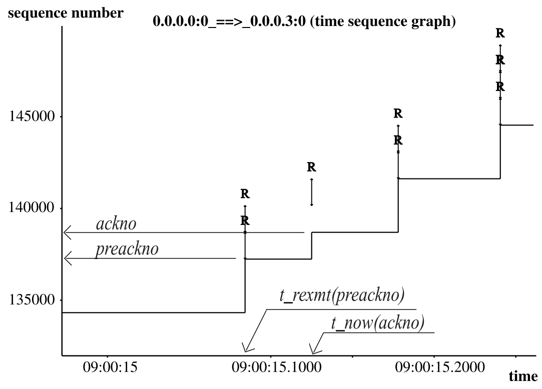

Fig. 4 The parameters for STO detection using RTT.

is cleared, the receiver acknowledges these segments one after that other, so that these acknowledgments arrive at the sender within a short time during go-back-N retransmission[9].

In order to detect STO, the proposed algorithm uses two thresholds for early arriving ACKs, (i) $RTT_{min}/2$ after the first unacknowledged segment retransmission as per Allman, and (ii) the time Δ during go-back-N retransmission as follows:

$$\begin{aligned}\Delta &= \texttt{t\_now(ackno)} - \texttt{t\_rexmt(preackno)} \\ &\le \alpha \times RTT_{min}/2 \qquad (1)\end{aligned}$$

where `t_now(ackno)` and `t_rexmt(preackno)` mean the time of the last acknowledgment arrival and that of the retransmitted segment, which have sequence number "`ackno`" and "`preackno`", respectively. **Figure 4** shows the relationship between `t_now(ackno)` and `t_rexmt(preackno)`. The X-axis and Y-axis plot the time and the sequence number, respectively. The bold lines with "R" and thin lines show the retransmitted segments and the acknowledgments, respectively. Moreover, the segment and the acknowledgment have the relationship `ackno` − `preackno` = `rxmt_size` where `rxmt_size` means the size of the retransmitted segments. α is a constant.

Figure 5 shows an example of the STO detections based on the proposed algorithm using an early arriving ACK. The parameter α in

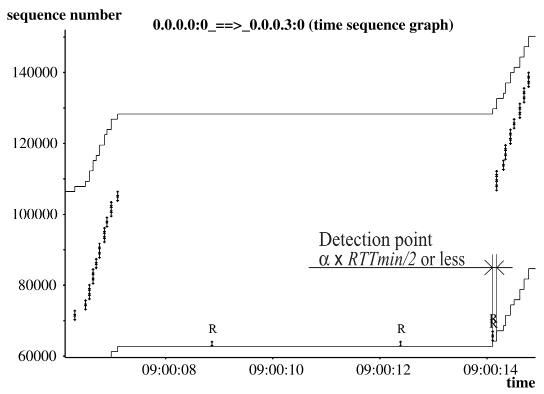

Fig. 5 STO detection using the early arriving ACK.

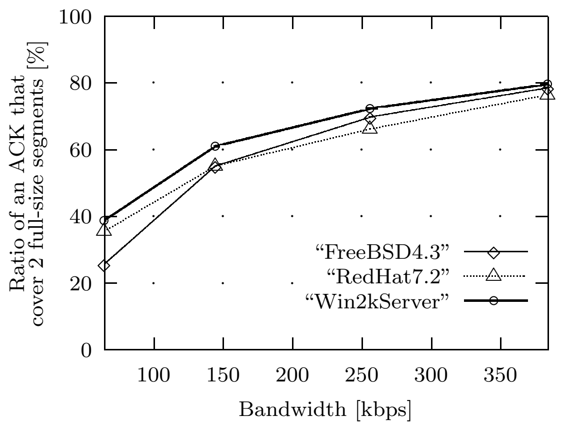

Fig. 6 Relationship between bandwidth and an acknowledgment cover 2 full-seize segments.

Equation (1) is given by

$$\alpha = 1 + (\texttt{ackno} - \texttt{preackno})/\texttt{window\_size}$$

where window_size means the receiver's maximum window size. The simulation model (*ns*2 version 2.1b9) is based on Fig. 2; we used $MSS = 1,460$ bytes and window_size = 64 kbytes. The segments retransmitted according to the exponential backoff algorithm arrive at 9:00:8.9 and 9:00:12.4, and the first acknowledgment following the RTO arrives at 9:00:14.1 in Fig. 5. The sequence number of this acknowledgment covers the retransmitted segment. The proposed algorithm initially cannot identify whether the acknowledgment is sent in response to the original segment or the retransmitted segment. The next unacknowledged segments are then retransmitted by the slow start algorithm. Just after retransmission, the second acknowledgment arrives after 100 ms ($< RTT_{min}/2$), so the proposed algorithm detects STO and reverts to the original cwnd and ssthresh. As a result, a sender with the proposed algorithm can avoid unnecessary retransmission; it avoids the segment retransmission caused by the slow start algorithm and the attendant throughput degradation.

4.2 STO Detection Based on ACKs Covering 2 or More Segments

When the wireless link recovers after experiencing excessive delay jitter or receiver's movement outside the service area, a series of queued segments arrives at the receiver. If the wireless link provides sufficient bandwidth, the receiver sends acknowledgments that cover 2 full-size segments in accordance with delayed ACK[10]. The delayed ACK mechanism only specifies that an acknowledgment is generated for at least every second full-size segment or within 500 ms from the arrival of the first unacknowledged segment.

For open source operating systems such as FreeBSD and Linux, it is easy to check the delayed ACK implementation, but Windows OS is a closed proprietary operating system. We used a WCDMA emulator[16] to determine what percentage of acknowledgments cover 2 full-size segments for FreeBSD 4.3, RedHat 7.2, and Windows2000 Server. **Figure 6** shows the relationship between bandwidth and the ratio of acknowledgments that cover 2 full-size segments based on the simulation model shown in Fig. 2. The simulation results are the average of 10 trials of 1 MB data transmission under 5% block error rate wireless link conditions. It is shown that the number of acknowledgments that cover 2 full-size segments in a connection increases gradually with the bandwidth. If the mobile station is connected to the BS via a 384 kbps wireless link, it reaches 80%. As a result, it is consider that the proposed algorithm can detect STO events with sufficient accuracy. The proposed algorithm can, moreover, detect an STO if the acknowledgment covers some full-size segments in the same way as an acknowledgment covers 2 full-size segments. If the acknowledgment covers the arrival of 2 or more full-size segments, the proposed algorithm reverts to the original cwnd and ssthresh. It then uses the fast retransmit/recovery algorithm as per the conventional algorithm if it receives duplicate ACKs.

Figure 7 shows an example of STO detection based on the proposed algorithm using an acknowledgment that covers 2 full-size segments caused by the delayed ACK. The segments re-

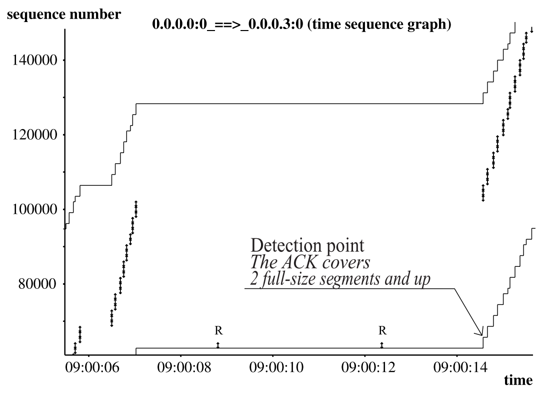

Fig. 7 STO detection using an acknowledgment that covers 2 full-size segments.

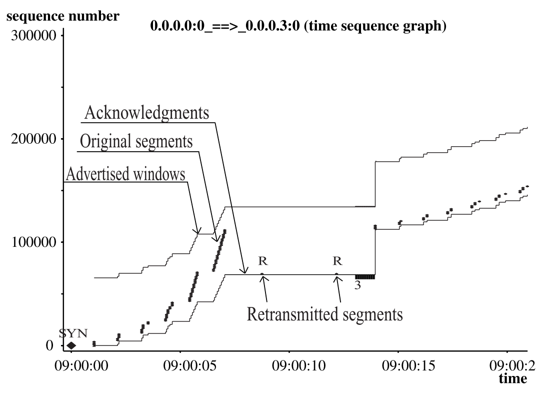

Fig. 8 An example of duplicate ACK arrival after an RTO using the conventional algorithm.

transmitted according to the exponential backoff algorithm arrive at 9:00:8.9 and 9:00:12.4, and the first acknowledgment to cover 2 full-size segments following the RTO arrives at 9:00:14.6 in Fig. 7. This acknowledgment is the response to the original segment, so the sender detects STO and reverts to the original `cwnd` and `ssthresh`. As a result, a sender running the proposed algorithm can avoid unnecessary retransmission without experiencing the segment retransmission demanded by the slow start algorithm and the attendant throughput degradation.

4.3 Extended STO Detection Based on Duplicate ACK

This section describes an extension of the proposed STO detection algorithm to handle the duplicate ACKs that follow an RTO. The duplicate ACKs are caused by the expiration of ARQ retransmission; link layer transmission does not succeed for a period of time or for a number of transmissions. The conventional STO detection algorithms [3),6)~8)] have no function to detect these ACKs, and they revert to conventional retransmission, even if only one segment is lost while the others have been accepted by the receiver. Moreover, these algorithms do not work well even if the receiver use the SACK option and the duplicate ACKs contain SACK blocks. In this case, a sender with the conventional algorithm, directly enters the conventional avoidance [10)] phase, not the slow start phase, after the RTO, so that it needs additional time to cut `cwnd` to half of its previous value which leads to a throughput degradation.

The proposed algorithm monitors duplicate ACKs following the RTO, and controls both `cwnd` and `ssthresh`, because these acknowledgments are the response to the original segments. When the number of duplicate ACKs reaches the duplicate acknowledgment threshold (DupThresh), the proposed algorithm enters the fast retransmit/recovery algorithm directly. DupThresh is currently specified as the fixed value of three for the fast retransmit algorithm [10)]. Moreover, the proposed algorithm with the SACK option can retransmit lost segments accurately. By using acknowledged sequence numbers and received sequence numbers as SACK blocks, the SACK information shows the unacknowledged holes; this allows the proposed algorithm to respond effectively after an RTO similar to conventional SACK recovery.

Figure 8 shows an example of an RTO with segment loss behavior of the conventional algorithm. The symbol "3" shows the arrival of the third duplicate ACK at the sender. In this case, the sender does not transmit any segments even if it receives duplicate ACKs following the RTO. It then enters the congestion avoidance phase directly, so that it takes a long time to cut `cwnd` to half of its previous value. On the other hand, **Figs. 9** and **10** show an example of an RTO with the segment loss behavior of the proposed algorithm using duplicate ACK monitoring. It reverts to the original `cwnd` and `ssthresh`, and enters the fast retransmit/recovery algorithm. As a result, the sender starts the congestion avoidance phase at half of the previous value when it receives the acceptable ACK, i.e. advances the acknowledged sequence number. It is clear that the proposed algorithm can avoid unnecessary `cwnd` reduction and throughput degradation more often than the conventional algorithm.

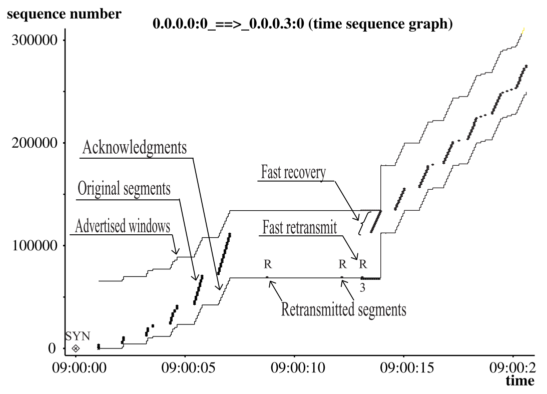

Fig. 9 An example of duplicate ACKs arrival after a RTO using the proposed algorithm.

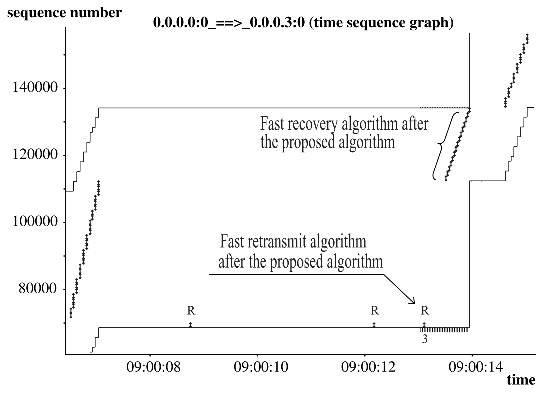

Fig. 10 An example of the fast retransmit/recovery algorithm after an RTO using the proposed algorithm.

4.4 STO Detection Steps

Figure 11 shows the STO detection steps in the proposed algorithm using acknowledgment monitoring and congestion control.

At the beginning, the sender stores `cwnd`, `ssthresh`, and the highest sequence number (`send_high`) when the sender has an RTO. According to the exponential backoff algorithm, an unacknowledged segment is retransmitted (step 1). When the first acceptable ACK or duplicate ACK following the RTO arrives at the sender, the sender chooses an action in step 2. Step 3 is chosen if the Nth ($N > 1$) acceptable ACK or duplicate ACK arrives at the sender.

The duplicate ACKs in step 2 (a) and 3 (a) are the response to an original segment that is one of the outstanding segments. When the 3rd duplicate ACK arrives, the sender enters the fast recovery/retransmit algorithm (Section 4.3). Steps 2 (b) and 3 (b) evaluate an early arrival ACK as per Allman's report (Sec-

```
Step 1. RTO timer expires:
    Store send_high (highest sequence number),
        cwnd_ ← cwnd (congestion window size),
      ssthresh_ ← ssthresh (slow start threshold)
    Retransmit the 1st unacknowledged segment

Step 2. 1st acknowledgment arrival:
  Evaluate the acknowledgment sequence number
  (a) Duplicate ACK
      Proceed step 3
  (b) Advance the 1st unacknowledged segment
      Evaluate the time $\Delta$ between the retransmitted
      segment and this acknowledgment
        IF $\Delta < RTTmin/2$ THEN Proceed step 4
        ELSE Proceed step 3
  (c) Above the 1st unacknowledged segment and
      below the value of send_high
      Proceed step 4
  (d) Acknowledge all segments up to send_high
      IF DSACK carries THEN Proceed step 4
      ELSE Proceed step 5

Step 3. Nth acknowledgment arrival:
  Evaluate the acknowledgment sequence number
  (a) Duplicate ACK
      IF 3rd duplicate ACK THEN
       Restore cwnd ← cwnd_, ssthresh ← ssthresh_
        Shift fast retransmit/recovery
      ELSE Remain step 3
  (b) Below the previous unacknowledged segment
      Evaluates the time $\Delta$
        IF $\Delta < \alpha RTTmin/2$ THEN Proceeds step 4
        ELSE Retransmit next segments according
            to the slow start and Proceeds step 3
  (c) Above the previous unacknowledged segment
      and below the value of send_high
      Proceed step 4
  (d) Acknowledge all segments up to send_high
      Proceed step 5

Step 4. Label spurious:
    Recover cwnd ← cwnd_, ssthresh ← ssthresh_
    Release send_high and transmit new segments

Step 5. Label spurious with a possibility of segments
loss:
    Recover ssthresh ← ssthresh_/2,
        cwnd ← ssthresh + $3MSS$
    Transmit new segments
```

Fig. 11 The proposed algorithm steps for the STO detection and congestion control.

tion 4.1). Step 2 (c) observes an acknowledgment that covers 2 or more full-size segments caused by delayed ACKs (Section 4.2). If the sender has multiple connections, all of which are equally and perfectly filled with segments, the sender may not receive acknowledgments that cover 2 or more full-size segments and so may shift to step 3 in accordance with the slow start algorithm. In the case of step 2(d), the sender reverts to the original `cwnd`, and `ssthresh` as in the fast retransmit algorithm using step 5 because of the indication of segment loss.

If the sender detects an STO event, it changes `cwnd_` and `ssthresh_` to the new `cwnd` and `ssthresh`, and transmits new segments in accordance with step 4. The fast retransmit/recovery algorithm is still available when the 3rd duplicate ACK arrives after step 4. Since the Reno fast retransmit algorithm leads to the retransmission of only a single data packet, NewReno[17] or the selective acknowledgement (SACK) option[11] is required to recover multiple packets that have been dropped in a single window. If a retransmission timeout occurs due to the loss of an entire window of segments and the acknowledgments are delayed, the proposed algorithm may experience a misunderstanding. However, it is designed to realize the same congestion window control as the conventional method without aggressive segment transmission, even if it experiences the false-positive condition (see Appendix A.1).

5. Performance Evaluation

In Section 4, we showed that the STO detection algorithm observes early arriving ACKs, acknowledgments that follow a delayed ACK that covers 2 or more full-size segments, and duplicate ACKs. In the case of STO detection, it reverts to the original `cwnd` and `ssthresh` to avoid unnecessary retransmission and throughput degradation as a form of congestion window control. However, the proposed and conventional algorithms show improvement only for corner cases. That is, if an STO does not occur or duplicate ACKs do not arrive, these algorithms are not triggered and so provide no improvement in the connection's throughput. It is difficult to define a realistic wireless communication models that includes STO and duplicate ACK arrival for throughput evaluations[18]. Accordingly, this paper examined `cwnd` behavior in the face of an STO and the arrival duplicate ACKs arrival following an RTO.

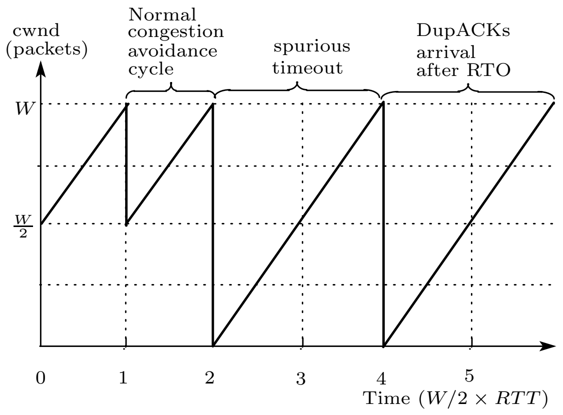

Fig. 12 Performance loss approximation of the normal TCP sender.

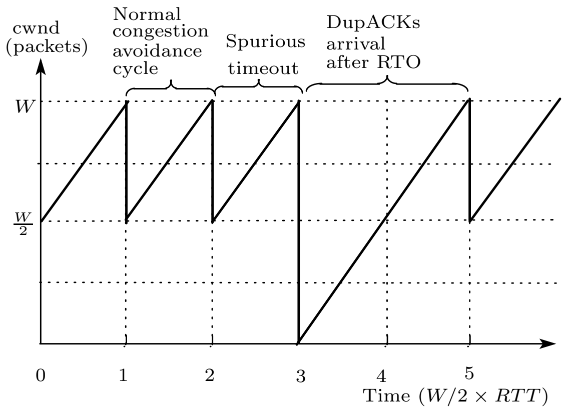

Fig. 13 Performance loss approximation of the sender with conventional STO detection algorithms.

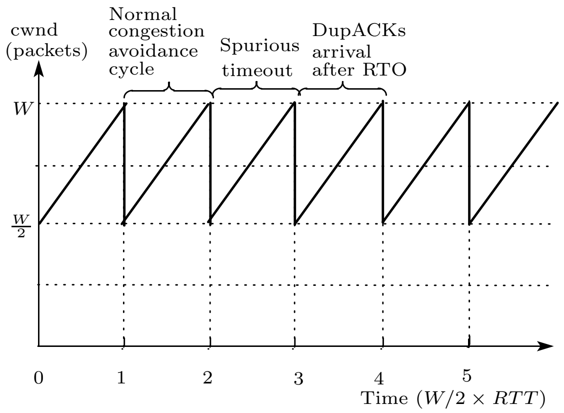

Fig. 14 Performance loss approximation of the sender with the proposed STO detection algorithm.

Figures 12, **13**, and **14** show `cwnd` behavior with bulk data transfer. The maximum window value is W and `cwnd` shifts $W/2$ following the definition of congestion avoidance[19]. We assume a network limited TCP connection in steady state that has only a single traffic source; packet loss is random with constant probability. Moreover, the increase in `cwnd` is taken as linear during the slow start phase to simplify the analysis. Given these assumptions, the figures show the performance loss evaluations caused by the decrease in `cwnd`. Without any STO detection algorithm, the sender has no function to alter `cwnd`, which leads to the throughput degradation seen in Fig. 12. With an STO detection algorithm (e.g., Eifel, DSACK, Allman's, and F-RTO), on the other hand, the sender can avoid the slow start phase if it detects the STO in Fig. 13. However, these algorithms does not have congestion window control mechanism that can handle the arrival of duplicate ACKs after an RTO, and so they suffer a throughput degradation. In order to avoid the slow start phase, the proposed algorithm has a detection mechanism that can handle RTO events and the arrival of duplicate ACKs. This means it can effectively avoid the unnecessary reduction in `cwnd` and maintain the window size even if STO occurs or duplicate ACKs are received following the RTO as shown in Fig. 14.

6. Conclusions

In this paper, we proposed an STO detection and congestion control algorithm. The proposed algorithm can detect an STO from either early arriving ACKs, an acknowledgment that covers 2 full-size segments, or the arrival of ACKs following the RTO. In response, it controls the congestion window to improve the throughput. *ns*2 simulations demonstrated its correct behavior under data transfers. Moreover, we evaluate its ability to minimize `cwnd` reduction and so enhance performance. Simulations showed that the proposed algorithm can maintain the congestion window size even if STO occurs or duplicate ACKs are received following an RTO. Therefore, it can avoid unnecessary retransmission and `cwnd` reduction, and effectively maintain the throughput.

Acknowledgments Acknowledgments The authors would like to thank Reiner Ludwig, Markku Kojo, and Daikichi Osuga for several useful discussions and comments.

References

1) Inamura, H., Montenegro, G., Ludwig, R., Gurtov, A. and Khafizov, F.: TCP over Second (2.5G) and Third (3G) Generation Wireless Networks, RFC3481 (2003).
2) 3GPP: 3G TS 25.322 v.3.5.0, RLC Protocol Specification (2000).
3) Ludwig, R. and Meyer, M.: The Eifel Detection Algorithm for TCP, RFC3522 (2003).
4) ANSI/IEEE Std 802.11: Wireless LAN medium access control (MAC) and physical layer specifications, 1999 Edition (1999).
5) ANSI/IEEE Std 802.11b: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, Higher-Speed Physical Layer Extension in the 2.4 GHz Band, 1999 Edition (1999).
6) Blanton, E. and Allman, M.: Using TCP Duplicate Selective Acknowledgement (DSACKs) and Stream Control Transmission Protocol (SCTP) Duplicate Transmission Sequence Numbers (TSNs) to Detect Spurious Retransmissions, RFC3708 (2004).
7) Allman, M. and Falk, A.: On the Effective Evaluation of TCP, *SIGCOMM Computer Communication Review*, Vol.29, No.5 (1999).
8) Sarolahti, P., Kojo, M. and Raatikainen, K.: F-RTO: An Enhanced Recovery Algorithm for TCP Retransmission Timeouts, *ACM SIGCOMM Computer Communication Review*, Vol.33, No.2 (2003).
9) Miyake, M., Inamura, H. and Takahashi, O.: An Efficient Spurious Timeout Detection Scheme for Mobile Communication, *Multimedia, Distributed, Cooperative and Mobile Symposium, DICOMO2003*, pp.205–208 (2003).
10) Allman, M., Paxson, V. and Stevens, W.: TCP Congestion Control, RFC2581 (1999).
11) Mathis, M., Mahdavi, J., Floyd, S. and Romanow, A.: TCP Selective Acknowledgment Options, RFC2018 (1996).
12) Floyd, S., Mahdavi, J., Mathis, M. and Podolsky, M.: An Extension to the Selective Acknowledgement (SACK) Option for TCP, RFC2883 (2000).
13) The Network Simulator – ns-2: http://www.isi.edu/nsnam/ns/ (ns 2.1b9).
14) Gurtov, A.: Effect of Delays on TCP Performance, *Proc. IFIP Personal Wireless Conference* (2001).
15) Hoe, J.: Improving the Start-up Behavior of a Congestion Control Scheme for TCP, *ACM SIGCOMM* (1996).
16) Inamura, H., Ishikawa, T., Atsumi, Y. and Takahashi, O.: Evaluation of Link ARQ and TCP over W-CDMA Network, *IPSJ Transactions*, Vol.43, No.12, pp.3859–3868 (2002) (in Japanese).
17) Floyd, S. and Henderson, T.: The NewReno Modification to TCP's Fast Recovery Algorithm, RFC2582 (1999).
18) Ludwig, R. and Katz, R.H.: The Eifel Algo-

```
  Transmitted   Received    ACK     cwnd   ss  cwnd_
  Segment       Segment     Sent         thresh

 1: 500-999     (data packet dropped)  4   44    -
 2: 1000-1499   (delayed)              4   44    -
 3: 1500-1999   (data packet dropped)  4   44    -
 4: 2000-2499   (delayed)              4   44    -
 5: (timeout)
 6: 500-999(R)  (delayed)              1    2    4
 7:             1000-1499   500        1    2    4
 8:             2000-2499   500        1    2    4
 9:              500-999(R) 1500       5   44    4
10: 2500-2999   2500-2999   1500       5   44    4
11: 3000-3499   3000-3499   1500       5   44    4
12: 3500-3999   3500-3999   1500       5   44    4
13: 1500-2000(R) 1500-2000  4000       5    2    4
14: 4000-3499   4000-4499   4500       2    2    4
15: 4500-3999   4500-4999   5000       2    2    4
...
```

Fig. 15 An example of an ACK covers 2 full-size segments arrival after a RTO.

rithm: Marking TCP Robust Against Spurious Retransmission, *SIGCOMM Computer Communication Review*, Vol.30, No.1 (2000).

19) Mathis, M., Semke, J., Mahdavi, J. and Ott, T.: The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm, *Computer Communications Review*, Vol.27, No.3 (1997).

Appendix

A.1 Misunderstanding in False-Positive Condition

If a retransmission timeout is caused by the loss of an entire window of segments and the acknowledgments are delayed, the proposed method may experience a misunderstanding. **Figure 15** shows a sequence graph on the sender and receiver sides. The first and last three columns show the data segment transmission and parameters on the receiver. The second and third columns show the data segment arrival and ACK transmission on the receiver, respectively.

In Fig. 15, the beginning and third of the four outstanding segments are lost, and the retransmitted segment raised by RTO is transmitted at the 6th line. Next, an acknowledgment that covers 2 full-size segments arrives at the sender at the 9th line. According to the STO detection procedure proposed in Section 4.2, the sender detects the STO and changes `cwnd_` to `cwnd`. After that it transmits the next three segments at the 10–12th lines even though the STO has not happened. When the third duplicate ACK arrives, the sender allows the transmission of the first unacknowledged segment at the 13th line, because of the release of `send_high` in accordance with step 4. The sender then reduces the `cwnd` and `ssthresh`, and recovers against the segment loss as in the usual congestion control procedure. If some outstanding segments are lost, it is better for the sender to use the proposed algorithm with the SACK [11] or NewReno [17] algorithm for effective segment loss recovery. Consequently, the proposed algorithm can realize the same congestion window control results as the conventional method without aggressive segment transmission, even if it experiences the false-positive condition.

(Received December 26, 2003)
(Accepted October 4, 2004)
(Online version of this article can be found in the IPSJ Digital Courier, Vol.1, pp.15–25.)

Editor's Recommendation

Wireless communications using TCP lead to experience spurious timeout (STO) problems due to an unexpected increase in round trip time (RTT). In this case, the TCP sender assumes that outstanding segments are lost, and it retransmits unacknowledged segments even if they don't need to be transmitted. It is important to suppress such unnecessary retransmission, because these retransmitted segments affect the throughput, and they are charged in packet cellular systems. So, the authors propose a spurious timeout detection algorithm effective in wireless communications, and confirm its performance improvement using simulations.

The STO detection algorithms have been proposed, and implemented in some OSs. However they are not sufficient to use for wireless communications because of the detection period's restriction and additional information's requirement in the TCP header. Therefore, the proposed algorithm in this paper based on both an ACK arrival time after a retransmission timeout and the sequence number monitoring only needs the sender side modification and avoids additional information. These features are useful for applications in packet cellular systems, so this algorithm can expect the utilization to mobile communications.

(Steering member of SIGMBL
Tadanori Mizuno)

Motoharu Miyake was born in 1973. He received the B.S. and M.S. degrees in electronic engineering from Tokyo University of Engineering, Tokyo, Japan in 1995 and 1997, respectively. He received the Ph.D. degree in electrical and electronic engineering from Tokyo Institute of Technology, Tokyo, Japan in 2000. Since 2000, he has been working in Multimedia Laboratories at NTT DoCoMo, Inc., Yokosuka, Japan. His current research interest is transport protocol for wireless communication. He is a member of IEICE.

Hiroshi Inamura was born in 1965. He received B.S. and M.S. degree in Keio University, Japan, in 1998 and 1990, respectively. He joined NTT in 1990 and working in the area of distributed systems and especially distributed file systems. From 1994 to 1995, He was a visited researcher in the Department of Computer Science, Carnegie Mellon University. From 1999 he worked for Multimedia Laboratories at NTT DoCoMo, Inc. His research interests are on transport protocol issues and their solutions for wireless. He is a member of IPSJ, IEICE and ACM.

Osamu Takahashi was born in 1951. He received the B.E. and the M.E. degrees from Hokkaido University, Sapporo, Japan, in 1973 and 1975, respectively. He received the Ph.D. degree from Shizuoka University, Shizuoka, Japan, in 2003. From 1975 to 1999, he was with NTT. From 1999 to 2004, he was with NTT DoCoMo, Inc. working on protocols and services of mobile multimedia communications. He is currently a professor of Future University-Hakodate, engaging in the research on mobile computing, ubiquitous networking and internet protocols.