Vol. 47 No. 2 IPSJ Journal Feb. 2006

Regular Paper

A TCP-Aware Link Layer Protocol Based on Selective-repeat ARQ with No Resequencing

Toshihiro Shikama,[†] Tatsuji Munaka,[†] Takashi Watanabe[††] and Tadanori Mizuno[††]

A link layer protocol based on SR (Selective-Repeat) ARQ (Automatic Repeat reQuest) is required to achieve high performance over a lossy and large delay link. In spite of its effectiveness, SR ARQ has problems of large resequencing delay and bursty packet output due to the resequencing function. To mitigate these problems, this paper proposes a scheme where the resequencing is not performed by a receiver of SR ARQ, but by TCP end to end. The proposed scheme has advantages of scalability with regard to link bandwidth, the number of TCP connections on the link, and the size of the TCP window. It also preserves the end-to-end semantics of TCP and requires no modification to the existing TCP. To suppress unnecessary retransmissions by TCP, a sender of SR ARQ is aware of retransmitted TCP data packets and drops duplicate ACKs due to out-of-order packets, which are caused by the lack of resequencing by SR ARQ. The effectiveness of this proposed scheme is evaluated by simulations, which show that it attains high performance in comparison with other schemes.

1. Introduction

In broadband networks that employ lossy and large delay links, such as satellite or terrestrial radio networks, a retransmission protocol as well as forward error correction (FEC) is generally required. In the case of wireline IP networks, retransmissions are performed by an end-to-end protocol such as TCP or locally by a link layer protocol. If transmission errors over a link are rare, end-to-end retransmissions by TCP are capable of recovering packet losses. However, TCP assumes that losses of packets occur as a result of congestion of a network. It decreases its congestion window after it retransmits packets, and as a result the throughput of TCP temporarily decreases [1)]. Because of this effect, end-to-end retransmissions by TCP incur degradation of throughput when it is used for a network employing a lossy and large delay link.

For a network including such a link, an approach that employs local retransmissions over the link and hides losses of packets from end-to-end TCP is able to achieve high performance [2)]. For example, in the protocol stack of W-CDMA, Acknowledged Mode (AM) and Unacknowledged Mode (UM) are provided by Radio Link Control, which is a sub-layer of the link layer (layer 2) [3),4)]. RLC segments a Service Data Unit (SDU) from the upper layer into one or more Protocol Data Units (PDUs). AM performs retransmissions of PDUs to recover errors over a radio channel, while DM does not recover lost PDUs. In order to obtain high performance for the case where TCP is employed end-to-end, local retransmissions by AM over a radio channel are required.

In this paper we assume a link layer protocol that performs local retransmissions over a lossy link. Among such local retransmission protocols applicable to a large delay link, SR (Selective-Repeat) ARQ (Automatic Repeat reQuest) is superior to other ARQ protocols, since it retransmits the minimum number of packets that are actually lost over the link. However, this SR ARQ requires resequencing of packets at a receiver to preserve the order of packets. This resequencing causes the problems described below.

Figure 1 shows an example of a conventional sequence of TCP data packets over a link layer protocol based on SR ARQ. SR ARQ retransmits a packet that encountered transmission errors over a link. SR ARQ retains packets that are correctly received after the lost packet to preserve the original order of packets. Although the efficiency of SR ARQ is excellent, this resequencing incurs a large delay for each retained packet. There is also another problem: when a retransmitted packet is successfully received, all retained packets waiting for the retransmitted one are released and forwarded to outgoing

† Mitsubishi Electric Corporation
†† Shizuoka University

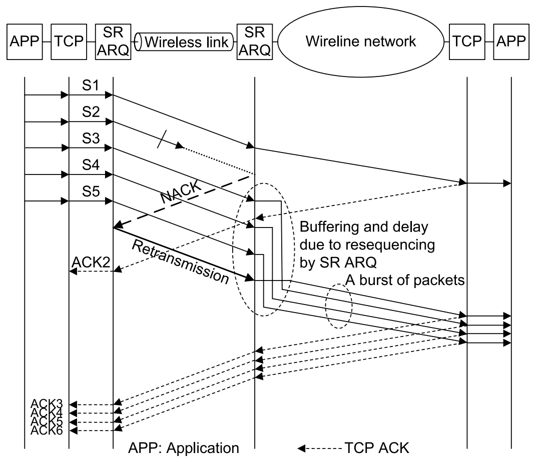

Fig. 1 Example of a conventional sequence of TCP data packets, where a link layer protocol based on SR ARQ is employed.

links at the same time. Therefore, SR ARQ produces bursts of packets, which cause undesirable effects such as buffer overflow at a forwarding node and QoS degradation of other flows sharing bandwidth. The problems described above become significant if the product of the delay and the bandwidth of a link increases, since the number of outstanding packets has to be large enough to fully utilize the link bandwidth. A typical case is a high-speed satellite link, but recent terrestrial wireless links, whose bandwidth is increasing rapidly, also need high-performance ARQ and will have the same problems. This paper deals with the problems associated with general SR ARQ protocols on the assumption that SR ARQ will be increasingly employed as the transmission rate of terrestrial wireless and satellite communications becomes large.

Hitherto, link layer protocols have been designed on the principle that the order of packets has to be preserved. However, TCP was originally designed for networks providing datagram services, where the order of IP packets is not guaranteed. TCP is capable of reordering packets received out of order. If the reordering capability of TCP can be employed for out-of-order packets derived from packet losses over a lossy link, resequencing by a link layer protocol is not required. With this motivation, we propose a link layer protocol based on SR ARQ with no resequencing, where the resequencing of packets is performed by TCP end to end. As for link layer protocols, this paper assumes no specific SR ARQ protocol; examples of the SR ARQ protocols supposed are RLC in the case of W-CDMA and HDLC with SREJ for general wireless or terrestrial lines 5). In the simulation we employ SSCOP because of its large sequence number space and simple protocol mechanism 14).

The rest of this paper is organized as follows: Section 2 discusses related work concerning local retransmissions. Section 3 proposes a scheme to mitigate problems caused by the resequencing by the link layer protocol. Section 4 describes the simulation conditions and presents the results along with some discussion. Finally, our conclusions are presented in Section 5.

A unit of data is generally called a segment, a packet, or a frame, depending on the layer which handles it. In this paper we use the term "packet" for all the layers, to unify the terminology.

2. Related Work

To mitigate the above-mentioned problems, the PFRS (Per-Flow ReSequencing) scheme was proposed 6), where the resequencing is performed for each upper layer flow independently, while detection of lost packets and associated retransmissions are carried out on the basis of the overall flows in the same way as in the conventional SR ARQ. As the resequencing is carried out for each flow, the PFRS scheme suppresses the resequencing delay and the number of packets in a burst. The PFRS scheme requires no awareness with regard to upper layer protocols except identification of upper layer flows. It is reported that this PFRS scheme achieves excellent throughput in comparison with the conventional full resequencing scheme, and can limit the size of a burst to the TCP window size 6). However, if a large number of upper layer flows are multiplexed over SR ARQ, the PFRS scheme still requires a large number of resequencing buffers. Furthermore, if TCP employs a large window size by introducing the TCP window scale option 7) to achieve a large throughput, the PFRS scheme produces large bursts.

A number of ongoing research efforts are related to local retransmissions by a link layer protocol 8),16). Snoop 9) performs local retransmissions and conceals from TCP all packet losses over a wireless link. It keeps a local copy of each packet that has been forwarded over this link. A packet loss over the link is detected by means of duplicate TCP ACKs and a retained copy of the lost packet is retransmitted. Since Snoop merely performs retransmission of lost packets, packets are delivered out of order. In order to avoid unnecessary duplicate ACKs, which are caused by the out-of-order packets, Snoop drops duplicate ACKs. As previously mentioned, Snoop utilizes duplicate TCP ACKs to detect loss of packets. It generally takes longer to detect loss of packets than with a link layer protocol, which is optimized for local retransmissions. If a retransmitted packet is also lost, Snoop cannot use duplicate ACKs to detect this, and recovery using a TCP timeout is needed. Generally, a link layer protocol is efficient even in this case. Another problem of Snoop is that its retransmission mainly targets transfers from a base station to a terminal. For the reverse direction, local negative acknowledgments are needed to inform a TCP sender of packet losses over the wireless link. TCP with SACK is required for end terminals to realize the local negative acknowledgments.

In another study, called delayed duplicate acknowledgments (DDA)[10)], a link layer protocol with no resequencing is assumed. However, the link layer protocol itself is not described in detail; the main object is to obtain a Snoop-like protocol that is TCP-unaware. The study focuses on delaying acknowledgments at a TCP receiver in order to prevent unnecessary duplicate ACKs, which will incur a retransmission by TCP. Although this approach does not require TCP awareness at the link layer protocol, it is hard to specify the delay needed to make a decision as to whether duplicate ACKs - or more precisely the third and subsequent duplicate ACKs - should be released or not. Modification of the TCP receiver is also needed. This approach has a further defect of unnecessary delay in responding to packet losses due to network congestion in a wireline part of a network.

TULIP[11),12)] has been proposed as a link-layer-based solution tailored for half-duplex radio links. It is basically an SR ARQ protocol employing a sequence number and a bit vector, which is used for selective retransmissions. A bit of the vector is associated with each transmitted packet and identifies whether the packet has been received successfully or not. Since this scheme preserves the order of TCP data packets, duplicate ACKs are not generated; thus, TCP awareness is not needed. As this scheme completely preserves the order of all TCP data packets, it is basically equivalent to the conventional SR ARQ with resequencing of all packets. If the number of outstanding packets becomes large, TULIP has the same problems as the conventional SR ARQ, such as the "head of line" blocking described in Section 3.4.

All of the above approaches employ local retransmissions over a lossy link. However, except for the PFRS scheme, they focus mainly on links with small delays. It seems that their performance or protocol mechanisms are not suitable for links where the product of the delay and the bandwidth is large. In this paper, we propose a link layer protocol architecture suitable for lossy links with large delay and high bandwidth.

3. A Link Layer Protocol Based on SR ARQ with No Resequencing

3.1 Basic Architecture

The basic idea of the proposed scheme is shown in **Fig. 2**, where drawing of ACKs is omitted to improve readability. The proposed scheme does not perform resequencing at the SR ARQ receiver, while SR ARQ detects a packet loss and retransmits the packet in the conventional way. The resequencing of packets due to transmission errors is performed by TCP end to end. A receiver of TCP generally allocates receive buffers required for the resequencing and notifies the sender of TCP the volume of buffers available as an advertising window. A sender of TCP transmits packets at such a rate as not to exceed the capacity of either the congestion window or the advertising window.

3.2 A Problem due to the Lack of Resequencing

The link layer protocol with no resequencing causes out-of-order packets at the receiver of TCP. These packets incur duplicate ACKs,

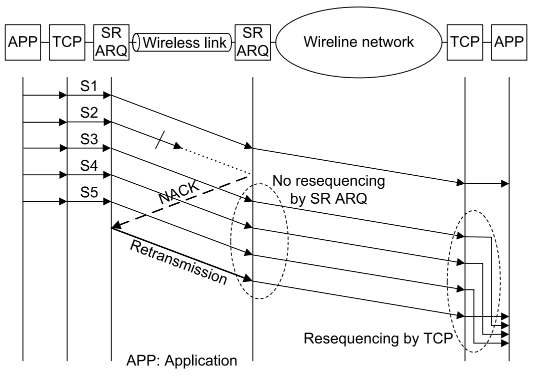

Fig. 2 Basic architecture of the proposed scheme.

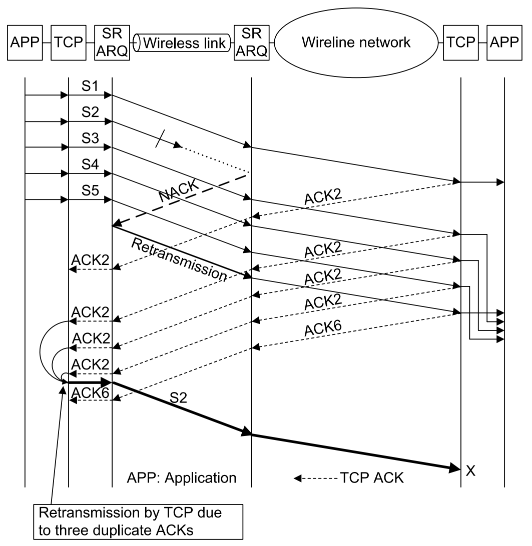

Fig. 3 A problem in the basic scheme.

as shown in **Fig. 3**. The receiver of TCP accepts out-of-order packets S3, S4, and S5, while waiting for a reception of packet S2. Each time it receives an out-of-order packet, it returns the same (duplicate) ACK2, which identifies the sequence number of the packet expected to be received next. When TCP receives a fixed number (normally three) of duplicate ACKs, it invokes the fast retransmit, which retransmits packet S2. As packet S2 has already been retransmitted by the link layer protocol, the retransmission by TCP is duplicated, and the second packet S2 is dropped by the receiver of TCP. Thus, unnecessary retransmission of a packet is performed by the duplicate ACKs. After the fast retransmit, TCP also performs fast recovery, where the size of the congestion window (cwnd) is reduced to half of its previous value. Although, after the fast recovery, the size of the congestion window increases in the congestion avoidance phase, its rate of increase is relatively slow, and the total throughput is suppressed on account of the insufficient size of the congestion window.

3.3 An Approach to Mitigating the Problem

One approach to mitigating the problem of unnecessary retransmission by duplicate ACKs may be to modify TCP so that it can be robust with respect to out-of-order packets. As explained before, this approach is adopted by a scheme called delayed duplicated acknowledgments (DDA) [10], but it requires modification of the existing TCP and still has issues to be addressed, as mentioned in Section 2. We propose a TCP-aware link layer protocol that does not require any modification of the existing TCP. Like Snoop, the proposed link layer protocol drops duplicate ACKs caused by loss of packets in the link layer, as shown in **Fig. 4**.

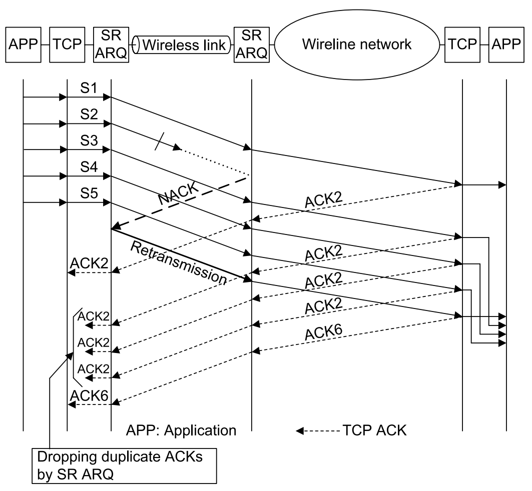

Fig. 4 Proposed scheme.

However, we have to preserve conventional fast retransmit by end-to-end TCP in provision for packet losses in a wireline part of the network. Therefore, duplicate ACKs should be dropped only in cases where they are caused by losses of packets over a lossy link on which SR ARQ is employed. Since SR ARQ is aware of which packets are lost and retransmitted, it is able to identify whether duplicate ACKs are caused by losses over the lossy link or by the wireline part of the network.

For this purpose, a sender of SR ARQ uses a table to record both the flow identification and the TCP sequence number for each retransmitted packet. The flow identification consists of the source and destination IP addresses, the source and destination port numbers, and the protocol ID. **Figure 5** shows an outline of the processing flow. Each time a sender of SR ARQ retransmits a TCP data packet, it stores the flow identification and TCP sequence number (TCP SN) of the packet in the table.

When a duplicate ACK is received, if both the flow identification and sequence number of ACK are recorded in the table, the ACK is dropped; otherwise, it is forwarded. If a packet loss occurs on account of congestion in the wireline part of the network, information concerning duplicate ACKs is not listed in the table, and they are forwarded normally. Then, con-

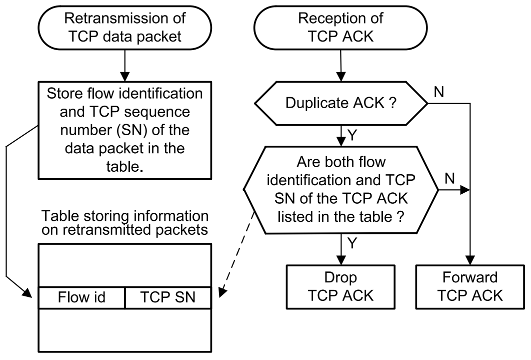
Retransmission of TCP data packet
Store flow identification and TCP sequence number (SN) of the data packet in the table.
Table storing information on retransmitted packets
Flow id
TCP SN
Reception of TCP ACK
Duplicate ACK ?
Y
N
Are both flow identification and TCP SN of the TCP ACK listed in the table ?
Y
N
Drop TCP ACK
Forward TCP ACK

Fig. 5 Outline of the processing flow.

ventional fast retransmit and fast recovery are invoked by end-to-end TCP. The above scheme assumes that the sender of SR ARQ is aware of the retransmission of TCP data packets.

In the proposed scheme, dropping of duplicate ACKs is terminated when a new ACK, which newly acknowledges outstanding TCP data packets, is received. There is a case where the maximum number of retransmissions is limited in SR ARQ: if a packet is not received correctly after the maximum number of retransmissions, it is not recovered by SR ARQ. There is also another case where a packet retransmitted by SR ARQ is correctly received but is lost in the wireline part of the network because of congestion. In these cases, a timeout at the TCP sender occurs and end-to-end retransmission is invoked. If this packet retransmitted by TCP is received correctly by the TCP receiver, a new ACK carrying an updated sequence number is returned to the TCP sender, and dropping of duplicate ACKs is terminated.

If a timeout occurs in TCP while SR ARQ is still trying to retransmit a TCP data packet, the TCP sender retransmits a duplicate TCP data packet. SR ARQ receives this packet and sends it normally. If the duplicate packet is received correctly while SR ARQ is still trying to retransmit the original packet, dropping of duplicate ACKs ends when a new ACK is received. If the duplicate packet is also lost due to transmission errors, SR ARQ will register the flow identification and the sequence number of the duplicate packet. Since this information has already been stored in the table, SR ARQ overwrites the same information. When either the original packet or the duplicated packet is correctly received, a new ACK is returned, and dropping of duplicate ACKs is terminated.

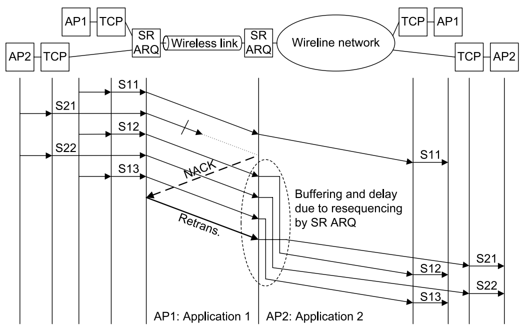

Fig. 6 The problem of HOL blocking.

3.4 Advantages of the Proposed Architecture

The proposed scheme has the advantage that it requires no resequencing buffers at the receiver of SR ARQ. In the conventional scheme, the required volume of resequencing buffers at the SR ARQ receiver increases significantly when one or more of the the following factors becomes large:

- The product of the bandwidth and the delay of a link
- The number of TCP connections over the link
- The TCP window size

Since the number of TCP connections and the size of the TCP window for each connection are not predictable, it is generally difficult to specify the required number of resequencing buffers. On the other hand, the proposed architecture has scalability in the resequencing, since the resequencing is performed at each TCP end independently. Another advantage is that the receiver of SR ARQ does not produce bursts of packets. A further advantage is the lack of HOL (head of line) blocking, which occurs when a number of upper layer flows are multiplexed over conventional SR ARQ, where the order of all packets is preserved [6]. **Figure 6** shows an example sequence of this problem. Two TCP connections are multiplexed over a lossy link where SR ARQ is employed. In the conventional scheme, since the order of all packets is preserved, packets S12 and S13 are delayed until packet S21 is received. In this case, packets S21 and S22 belong to a different TCP connection from packets S12 and S13. There is no need to keep packets S12 and S13 for resequencing. If the resequencing is not performed by SR ARQ, this HOL blocking never occurs, and packets S12 and S13 are forwarded without being delayed. In addition to the above-mentioned advantages, since the proposed scheme only drops duplicate ACKs, it still preserves the end-to-end semantics of TCP, including network congestion control.

4. Simulation

4.1 Simulation Configuration

Simulations were performed to evaluate the performance of the proposed scheme. We employed the network simulator ns-2 (ns-2.27) developed by the VINT project[13]. **Figure 7** shows the configuration of the simulation. Three nodes were connected by a wired link and a lossy wireless link. There were multiple TCP connections over these links. The simulation conditions are summarized in **Table 1**. The propagation delay and bit error rate of the wireless link were changed as listed in the table. A link layer protocol was employed over the lossy wireless link. For the actual protocol, we employed SSCOP (Service Specific Connection Oriented Protocol)[14], which realizes SR ARQ by a relatively simple mechanism.

The normal size of the TCP window that we used was 32 kbytes[4]. This value is too small for a single TCP connection to utilize the available link bandwidth fully, because of the large product of the delay and the bandwidth in the simulations. We assumed that multiple TCP connections exist over an SR ARQ connection and that the bandwidth of the link is fully utilized by the aggregated TCP connections. Packet errors occurred randomly following the bit error rate and the length of a packet.

The following five schemes were simulated and compared:

- Full resequencing
- PFRS
- SR ARQ with no resequencing (proposed scheme)
- Snoop
- No-link-layer ARQ (end-to-end retransmissions by TCP)

As Snoop is used for transmission from a base station to a terminal, the direction of all packet flows is from the wireline to the wireless, as in Fig. 7. The results of simulations are plotted in figures based on values averaged over 10 simulation runs.

4.2 Simulation Results

Figures 8 and **9** show comparisons of the above schemes where the numbers of TCP connections are 5 and 10, respectively. The bandwidth of a wireless link is assumed to be 5 Mb/s and its propagation delay is 0.25 sec, which corresponds to the value for a satellite link. These figures present the total throughput, which is the sum of the throughput of all TCP connections normalized by the bandwidth of the wireless link. In the simulation condition, the bandwidth of the link cannot be fully utilized in the case where the number of TCP connections is 5. The total throughput becomes large when the number of TCP connection is increased to 10.

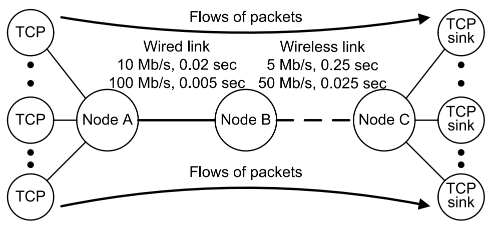

Fig. 7 Configuration of the simulated network.

Table 1 Simulation conditions.

Bandwidth of wireless link	5 Mb/s, 50 Mb/s
Propagation delay	0.025 s (50 Mb/s) 0.25 s (5 Mb/s)
Bit error rate	$10^{-8}, 10^{-7}$, $10^{-6}, 10^{-5}$
Type of bit error	Random
Link layer protocol	SSCOP
Polling interval of SSCOP	0.2 sec
TCP type	NewReno
TCP data packet length	1,460 bytes
TCP delayed acknowledgment	No Delay
Overhead by SSCOP	10 bytes
TCP window size	32 kbytes
No. of TCP connections	5, 10
Packet generation	Data always exists
Simulation time	60 sec
No. of simulation runs	10

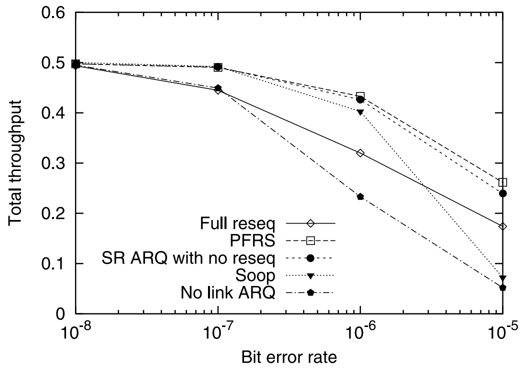

Fig. 8 Comparison of throughput. The number of TCP connections is 5, and the bandwidth and delay of the link are 5 Mb/s and 0.25 sec.

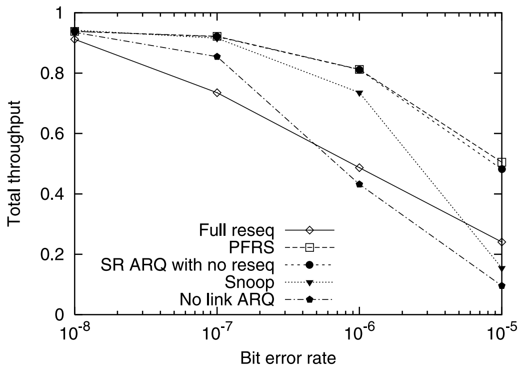

Fig. 9 Comparison of throughput. The number of TCP connections is 10, and the bandwidth and delay of the link are 5 Mb/s and 0.25 sec.

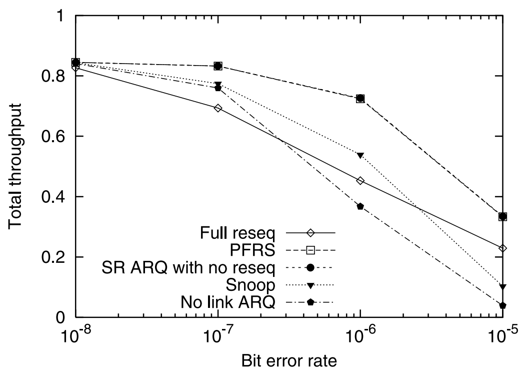

Fig. 10 Comparison of throughput. The number of TCP connections is 10, and the bandwidth and delay of the link are 50 Mb/s and 0.025 sec.

The performance of the PFRS scheme and the proposed scheme are the best among all the schemes. When the bit error rate is large (10^{-5}), the throughput of the proposed scheme is inferior to that of the PFRS scheme, but the difference is not significant. The same trend is observed in both Figs. 8 and 9, irrespective of the number of TCP connections.

When the bit error rate is 10^{-8}, the performance of Snoop is slightly better than that of the other schemes, since it has no link layer protocol overhead. However, as the bit error rate increases, its performance is degraded in comparison with the PFRS and the proposed schemes. The reason is considered to be that the retransmission capability of the SR ARQ-based link layer protocol is better than that of Snoop, which relies on duplicate ACKs and TCP timeouts. The conventional full resequencing scheme, which preserves the order of all packets over the link, shows low performance, since the delay due to resequencing and associated HOL blocking is large. The performance of the no link layer ARQ is also low, since retransmissions are performed by TCP end to end, and thus it takes time to retransmit lost packets and the congestion window is also decreased after each retransmission.

Figure 10 shows a comparison of the throughput for the case where the bandwidth of the wireless link is 50 Mb/s and its delay is 0.025 sec. The bandwidth of the wired link is changed to 100 Mb/s to avoid a bottleneck due to this link. Its delay is also changed to 0.005 sec so that the product of the delay and the bandwidth of the wireless link is almost the same as in the case shown in Fig. 9. Although the delay is small, the performance of the network is degraded when the bit error rate of the link becomes worse. The performance of the proposed scheme and the PFRS scheme is exactly the same in this case. Both schemes achieve excellent performance in comparison with other schemes. From these simulation results, it is clear that the proposed scheme achieves excellent performance close to that of the PFRS scheme. The superior performance of the proposed scheme is significant not only for long propagation links but also for high-speed wireless links.

4.3 Comparison with the DDA Scheme

As previously mentioned, the DDA scheme takes a similar approach to the proposed one, where the link layer protocol does not perform the resequencing. In the DDA scheme, a TCP receiver delays release of duplicate ACKs so that they do not return to a TCP sender unnecessarily. If the packet that is expected to be received next arrives at the receiver during this delay period, transmission of the duplicate ACKs is canceled. But the problem is the selection of the optimal delay value. If the delay is small, the probability of duplicate ACKs, which will lead to unnecessarily fast retransmit, becomes large; otherwise, recovery of packet losses in the wireline part of the network is delayed. **Figure 11** shows a comparison of the DDA scheme with the PFRS and proposed schemes, where the value of delay (d) is changed to 1.0 sec and 2.0 sec; the delay of a wireless link and its bandwidth are 0.25 sec and 5 Mb/s, respectively. When the value of the delay (d) is 1.0 sec, the performance of the DDA scheme is inferior to that of other schemes in the case of the large bit error rate. When the delay (d) is

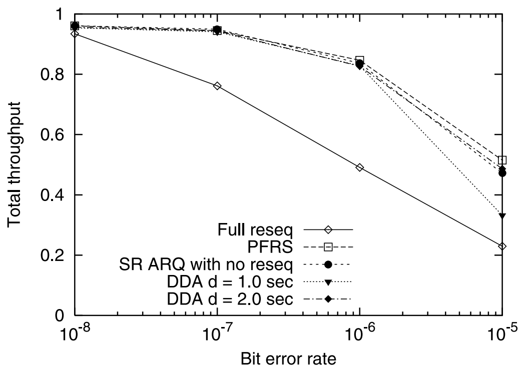

Fig. 11 Comparison with the DDA scheme. The number of TCP connections is 10, and the bandwidth and delay of the link are 5 Mb/s and 0.25 sec.

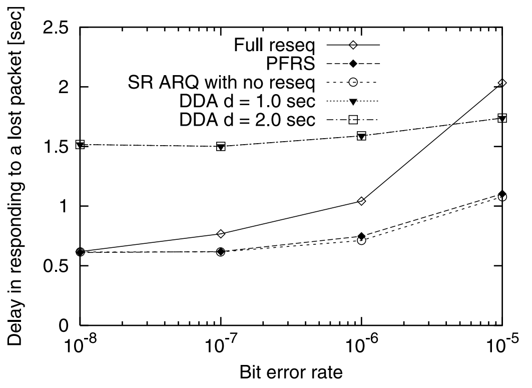

Fig. 12 Delay in responding to a lost packet. The number of TCP connections is 10, the bandwidth and delay of the link are 5 Mb/s and 0.25 sec, and the probability of packet loss in the wireline part of the network is 0.001.

2.0, the performance is comparable with that of the PFRS scheme and the proposed scheme.

Figure 12 shows the delay in responding to a packet loss that occurs randomly in the wireline part of the network. The probability of this packet loss is assumed to be 0.001. There are also packet losses over the wireless link. A response to a packet loss consists of both retransmission and a decrease in the size of the congestion window through either a fast recovery or a slow start. Recovery of packet losses in the wireline part of the network is delayed by transmission errors over the wireless link. It is clear that the DDA scheme takes longer to recover the packet losses than the PFRS and the proposed schemes. Since packet losses in the wireline part of the network occur because of congestion, rapid reactive control that reduces the size of the congestion window should be performed. The proposed scheme and the PFRS scheme are excellent in this regard, while the DDA scheme takes a long time to react to congestion.

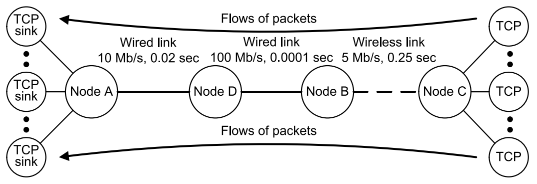
TCP sink
TCP sink
TCP sink
Flows of packets
Wired link 10 Mb/s, 0.02 sec
Wired link 100 Mb/s, 0.0001 sec
Wireless link 5 Mb/s, 0.25 sec
Node A
Node D
Node B
Node C
Flows of packets
TCP
TCP
TCP

Fig. 13 Configuration of the simulated network.

4.4 Performance in the Case where the Number of Buffers in a Forwarding Node is Limited

To compare the proposed scheme with the PFRS scheme, simulations were performed for the case where the number of buffers at a forwarding node is limited. **Figure 13** shows the configuration of the network simulated. In this figure, the direction of packet flows is from Node C to Node A. When the PFRS scheme is employed, bursts of packets are produced by Node B due to the resequencing and forwarded to Node D through a 100 Mb/s link. Node D transfers the packets to Node A through a 10 Mb/s link. Queuing of packets occurs at Node D because of the difference in link bandwidth. Simulations were performed for cases where the number of buffers (b) from Node D to Node A was changed to 50, 20, 18, 15, and 10. As the proposed scheme does not produce bursts of packets, the performance is the same irrespective of the number of buffers. The PFRS scheme produces bursts of packets, where the number of packets in a burst is limited by the TCP window size[6]. Since the size of the window was 32 kbytes in the simulations, up to 22 packets were forwarded at the same time and queued at Node D. If the number of buffers is smaller than the window size, packet losses are likely to occur. **Figure 14** shows the results of the simulations. It is observed that the throughput of the PFRS scheme is degraded as the number of buffers becomes small.

Figure 15 shows the number of resequencing buffers occupied by the receiver of SR ARQ, where the number of buffers in Node D is unlimited and the number of TCP connections N_{TCP} is changed to 5 and 10. As the bit rate and the number of TCP connections N_{TCP} increase, the number of resequencing buffers becomes large in the PFRS scheme. Since the proposed scheme does not perform resequenc-

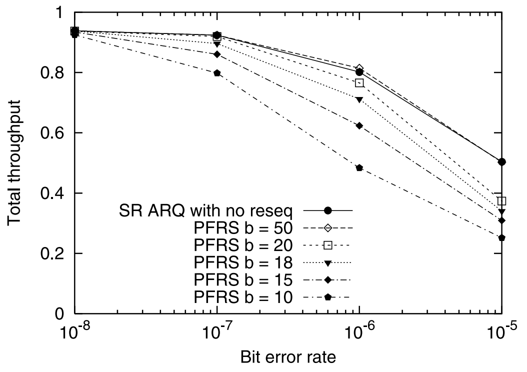

Fig. 14 Comparison of the throughput in the PFRS scheme and the proposed scheme when the number of buffers at a forwarding node is limited.

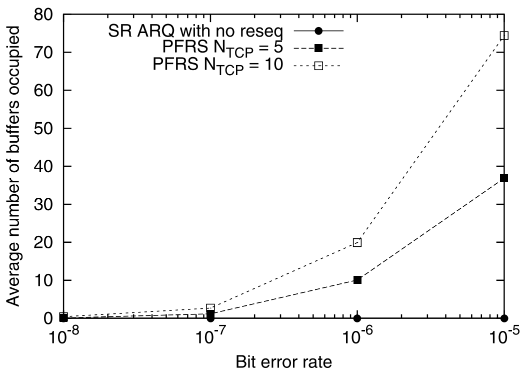

Fig. 15 Average number of resequencing buffers occupied.

ing, no buffer is occupied for all cases.

Figures 14 and 15 clearly show the advantages of the proposed scheme over the PFRS scheme in the case where the number of buffers at a forwarding node or at the receiver of SR ARQ is limited.

4.5 Spurious Timeouts and Recovery of the Congestion Window

As previously mentioned, the throughput of the proposed scheme is slightly inferior to that of the PFRS scheme in the case where the bit error rate is high. This difference is due to the following reasons:

- In the case of the proposed scheme, spurious timeouts of TCP are more likely to occur than in the PFRS scheme when the bit error rate is high.
- Recovery of the congestion window in the proposed scheme is slower than in the PFRS scheme when a TCP timeout occurs.

4.5.1 Spurious Timeouts

Table 2 shows the average number of timeouts that occurred on one TCP connection during the simulation time (60 sec). In this table "No reseq." represents the proposed scheme. It can be observed that spurious timeouts occur more frequently in the proposed scheme than in the PFRS scheme. The reason timeouts are likely to occur in the proposed scheme is considered to be as follows. Table 2 also presents the average round-trip time (RTT) measured by TCP during simulation runs. From this table, we can see that the measured RTT values for the proposed scheme are smaller than those for the PFRS scheme when the bit error rate is high. In the simulations, TCP always performs one RTT measurement at a time[15]. When resequencing is performed by the TCP receiver after a retransmission by SR ARQ, a single ACK acknowledging multiple TCP data packets is returned to the TCP sender. Measurement of RTT is likely to be terminated at this moment, and the next RTT measurement is started when the next new TCP data packet is sent. This new packet is unlikely to encounter resequencing, since there are no or few remaining outstanding packets preceding this packet. The measured RTT value tends to be smaller than the normal RTT value. This leads to a small retransmission timer value, which is considered to be the cause of the frequent timeouts.

Table 2 Average number of timeouts on one TCP connection and the average RTT value measured by TCP, where the number of TCP connections (N_{TCP}) is 10.

Bit error rate	Scheme	Timeouts	RTT [sec]
10^{-6}	PFRS	0.38	0.617
	No reseq.	0.56	0.593
10^{-5}	PFRS	0.41	0.955
	No reseq.	0.59	0.804

4.5.2 Recovery of the Congestion Window after a Timeout

Figure 16 shows an example of a congestion window (cwnd) change made by the proposed scheme when the bit error rate is 10^{-6} and the number of TCP connections is 5. **Figure 17** also shows an example of the same change made by the PFRS scheme in the same conditions. In both figures, the configuration of the simulated network is the same as in Fig. 13, except that Node D has unlimited buffers. In Fig. 17, one spurious timeout occurs, while two spurious timeouts occur in Fig. 16. A spurious timeout by TCP is likely to occur while a lost packet is being retransmitted by the link layer protocol. If a timeout occurs, the size of the conges-

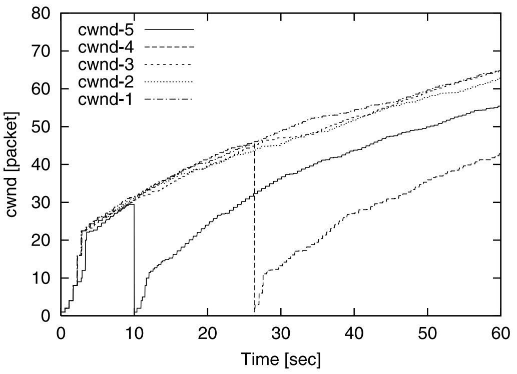

Fig. 16 Example of a cwnd change made by the proposed scheme. The bit error rate is 10^{-6}, the number of TCP connections is 5, and the bandwidth and delay of the link are 5 Mb/s and 0.25 sec.

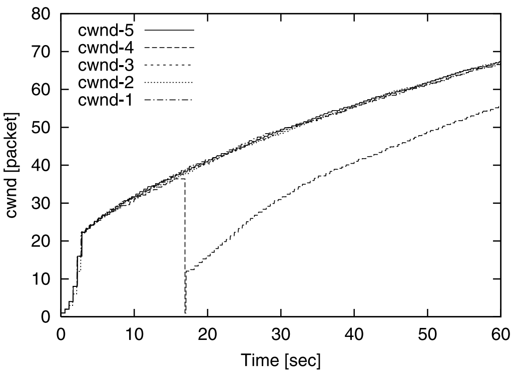

Fig. 17 Example of a cwnd change made by the PFRS scheme. The bit error rate is 10^{-6}, the number of TCP connections is 5, and the bandwidth and delay of the link are 5 Mb/s and 0.25 sec.

tion window (cwnd) is reduced to 1 and a slow start is invoked. As shown in the figures, the recovery of the congestion window in the proposed scheme is much slower than in the PFRS scheme. This causes the throughput of the proposed scheme to be small.

In the PFRS scheme, multiple in-sequence TCP data packets collectively arrive at the TCP receiver after the resequencing by SR ARQ. Multiple new ACKs are then returned to the TCP sender. These new ACKs are likely to arrive at the TCP sender collectively after a spurious timeout. Since update of cwnd is performed each time a new ACK is received, the congestion window increases rapidly in the PFRS scheme. **Figure 18** shows the changes in the sequence numbers of ACKs received by the TCP sender in the PFRS scheme. The box

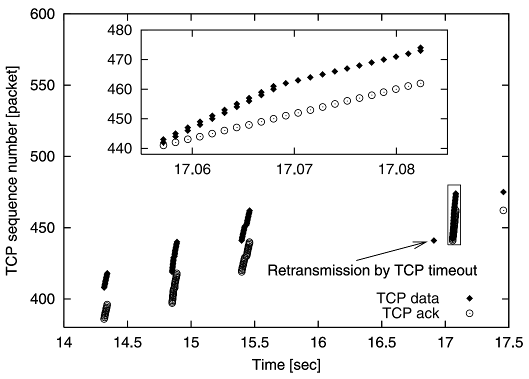

Fig. 18 Changes in TCP data and ACK sequence numbers in the case shown in Fig. 17.

in this figure shows details from 17.055 sec to 17.65 sec. The figure also presents the sequence numbers of TCP data packets sent by the TCP sender. In the simulations, a sequence number is assigned to each TCP data packet, and its value is incremented by 1 each time a new TCP data packet is sent. A retransmission by a TCP timeout occurs at around 16.9 sec. After this moment, 22 new ACKs are received from 17.057 sec to 17.083 sec. One or two data packets are sent each time a new ACK is received; the gap in sequence numbers between a received ACK and a sent data packet corresponds to the congestion window (cwnd). We can see that recovery of the congestion window is performed rapidly during this period.

On the other hand, in the proposed scheme, when the resequencing is performed by the TCP receiver after a retransmission by SR ARQ, only a single new ACK that acknowledges multiple TCP data packets is returned to the TCP sender. Update of cwnd is performed only once, when a lost packet is recovered by the link layer. This is why recovery of the congestion window in the proposed scheme is slower than in the PFRS scheme.

4.6 Bursts of Packets Sent by TCP

In the proposed scheme, as mentioned above, when resequencing is performed by the TCP receiver after retransmissions by SR ARQ, a single new ACK that acknowledges multiple TCP data packets is sent to the TCP sender. When the TCP sender receives the ACK, it can send a number of new TCP data packets at the same time. We call a group of packets sent by TCP at the same time a burst. Since packets whose number is equal to the TCP window size become outstanding during one RTT period, the number of new packets in a burst can also take

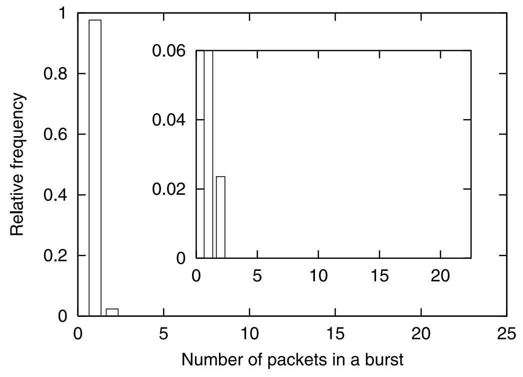

Fig. 19 Distribution of bursts at the TCP sender in the PFRS scheme. The bit error rate is 10^{-5} and the number of TCP connections is 10.

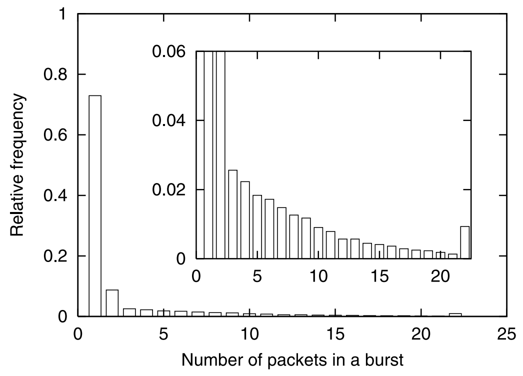

Fig. 20 Distribution of bursts at the TCP sender in the proposed scheme. The bit error rate is 10^{-5} and the number of TCP connections is 10.

any value up to or including the TCP window size.

Figures 19 and **20** show the distribution (relative frequency) of bursts at the TCP sender in the PFRS scheme and the proposed scheme, respectively, where the bit error rate is 10^{-5} and the number of TCP connections is 10. The box inside each figure presents details of small frequency values. Since the TCP window size is 32 kbytes and the length of a packet is 1,460 bytes, up to 22 packets can be outstanding. From Fig. 20, we can see that the proposed scheme produces bursts consisting of up to 22 packets. On the other hand, the PFRS scheme produces bursts of up to 2 packets, which are mainly sent during the slow start phase. In the simulation, a node accepting the bursts has enough buffers, but there is a possibility of packet losses if the number of buffers in the node is limited.

One approach to mitigating this problem is to limit the number of packets sent simultaneously when a new ACK is received[17]. In the simulations, TCP has a parameter called "maxburst" to limit the number of packets that can be sent in response to a single new ACK, but simulations were performed without making this parameter effective. Another approach is to suppress bursts by employing TCP pacing, where packets of the TCP window size are dispersed over one RTT period[18]. Since SR ARQ drops duplicate ACKs, there is also a possibility that SR ARQ regenerates new ACKs when a new ACK that collectively acknowledges multiple TCP data packets is received. Further investigations are needed to evaluate the effectiveness of these approaches.

5. Concluding Remarks

In this paper, we have studied a link layer protocol based on SR ARQ, focusing on its resequencing function. Instead of reordering packets at the receiver of SR ARQ, we proposed an architecture where the resequencing of packets is done by TCP end to end. The proposed approach has the advantages of no buffer requirement and no burst generation at the receiver of SR ARQ. It also has no HOL blocking when multiple upper layer flows are multiplexed over SR ARQ. These advantages become significant for broadband wireless networks, where the product of the bandwidth and the delay is increasing rapidly. It should also be noted that the proposed scheme preserves the end-to-end semantics of TCP, including network congestion control.

The performance of the proposed scheme was compared with that of other schemes through simulations, and was found to be excellent in the case where the product of the link bandwidth and the delay is large. When the number of buffers at a forwarding node is limited, the proposed scheme achieves a higher throughput than the PFRS scheme, since it does not produce bursts of packets.

Although the PFRS scheme has the advantage of TCP-unawareness, it has problems of scalability with regard to the number of TCP connections, the link bandwidth, and the size of the TCP window. The proposed scheme has no such defects, but TCP awareness is needed instead. These two schemes are complementary. For packet flows whose protocols are unknown, the PFRS scheme should be applied. However, if the type of packet flows is known to be TCP and the number of buffers at a receiver of SR ARQ is limited or bursts of packets by SR ARQ are not allowed, the proposed scheme provides an promising solution.

As mentioned in Section 4.6, the proposed scheme still has possible ways of improving spurious timeouts, slow recovery of the congestion window and bursts of packets generated by TCP. Approaches for mitigating these issues, and their evaluations, are left for further study.

References

1) Stevens, W.: TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, RFC 2001 (1997).

2) Fairhurst, G. and Wood, L.: Advice to Link Designers on Link Automatic Repeat reQuest (ARQ), RFC 3366 (2002).

3) 3GPP: TS.25.322, Radio Link Control (RLC) Protocol Specification (2004).

4) Inamura, H., Ishikawa, T., Atsumi, Y. and Takahashi O.: Evaluation of Link ARQ and TCP over W-CDMA Network, *IPSJ Journal*, Vol.43, No.12, pp.3859–3868 (2002) (Japanese Edition).

5) International Organization for Standardization: Information Processing Systems — Data Communication High-level Data Link Control Procedure — Frame Structure, IS 3309, 3rd ed. (Oct. 1984).

6) Shikama, T. and Mizuno, T.: A Proposal of the Re-sequencing Scheme for the Selective-Repeat ARQ and Its Performance Evaluation, *Trans. IEICE*, Vol.J88-B, No.4, pp.718–727 (2005) (Japanese Edition).

7) Jacobson, V., Braden, R. and Borman, D.: TCP Extensions for High Performance, RFC 1323 (1992).

8) Montenegro, G., Dawkins, S., Kojo, M., Magret, V. and Vaidya, N.: Long Thin Networks, RFC 2757 (2000).

9) Balakrishnan, H., Seshan, S., Amir, E. and Katz, R.: Improving TCP Performance over Wireless Networks, *1st ACM International Conference on Mobile Computing and Networking* (*Mobicom*) (1995).

10) Vaidya, N., Mehta, M., Perkins, C. and Montenegro, G.: Delayed Duplicate Acknowledgments: A TCP-Unaware Approach to Improve Performance of TCP over Wireless, Technical Report 99-003, Computer Science Dept., Texas A&M University (1999).

11) Parsa, C. and Garcia-Luna-Aceves, J.J.: TULIP: A Link-Level Protocol for Improving TCP over Wireless Links, *Proc. IEEE WCNC'99*, pp.1253–1257 (1999).

12) Parsa, C. and Garcia-Luna-Aceves, J.J.: Improving TCP Performance over Wireless Networks, *Mobile Networks and Applications Journal*, Vol.5, No.1, pp.57–71 (2000).

13) VINT Project: Network Simulator ns-2. http://www.isi.edu/nsnam/ns/

14) ITU-T: Recommendation Q.2110: B-ISDN Signalling ATM Adaptation Layer — Service Specific Connection Oriented Protocol (SSCOP) (1994).

15) Paxson, V. and Allman, M.: Computing TCP's Retransmission Timer, RFC 2988 (2000).

16) Pentikousis, K.: Wired-Cum-Wireless Environments, *IEEE Communications Surveys*, Vol.3, No.4 (2000). http://www.comsoc.org/pubs/surveys/

17) Mathis, M., Mahdavi, J., Floyd, S. and Romanow, A.: TCP Selective Acknowledgment Options, RFC 2018 (1996).

18) Aggarwal, A., Savage, S. and Anderson, T.: Understanding the Performance of TCP Pacing, *Proc. IEEE INFOCOM*, Vol.3, pp.1157–1165 (2000).

(Received May 9, 2005)
(Accepted November 1, 2005)
(Online version of this article can be found in the IPSJ Digital Courier, Vol.2, pp.57–69.)

Toshihiro Shikama received his B.E and M.E. degrees from Tokyo Institute of Technology in 1974 and 1976, respectively. From 1984 to 1985, he was at University of Waterloo, where he received the MASc degree. Since joining Mitsubishi Electric Corp. in 1976, he has engaged in developing a computer network using a satellite channel, high speed ring type LANs, time division multiplexers, ATM equipment, a high speed IP switch, and network security systems. He is a member of IPSJ, IEICE, and IEEE Communications Society.

Tatsuji Munaka received the M.S. degree in mathematical science from Tokyo Denki University, Japan in 1986. He received the D.E. degree in science and engineering from Shizuoka University, Japan in 2003. He is with Information Technology R&D Center, Mitsubishi Electric Corporation, Kamakura, Japan. During 1999–2002, he was a researcher of Yokosuka ITS Research Center, Telecommunications Advancement Organization of Japan (TAO) and studied about reliable multicast protocol in ITS network. His current research interests include mobile systems, wireless systems and network security. He is a member of IPSJ and IEEE Computer Society.

Takashi Watanabe received the M.E. and D.E. degrees from Osaka University, Japan in 1984, and 1987, respectively. In 1987, he joined the Faculty of Engineering, Tokushima University as an assistant professor. In 1990, he moved to the Faculty of Information, Shizuoka University. He was a visiting researcher at University of California, Irvine in 1995. He is currently a professor of Faculty of Informatics, Shizuoka University. His interests include computer networks and distributed system, especially MAC/Routing protocols for ad hoc and sensor networks. He is a member of IPSJ, IEICE, IEEE Communications/Computer Society, and ACM SIGMOBILE. He is currently Chair of the special interest groups of mobile computing and ubiquitous communications (MBL) of IPSJ.

Tadanori Mizuno received the B.E. degree in industrial engineering from the Nagoya Institute of Technology in 1968 and received the Ph.D. degree in computer science from Kyushu University, Japan, in 1987. In 1968, he joined Mitsubishi Electric Corp. Since 1993, he is a Professor of Faculty of Engineering, Shizuoka University, Japan. He moved to the Faculty of Information, Shizuoka University in 1995. His research interests include mobile computing, distributed computing, computer networks, broadcast communication and computing, and protocol engineering. He is a member of IPSJ, IEICE, IEEE Computer Society, and ACM.