PAPER *Special Section on Advances in Ad Hoc Mobile Communications and Networking*

Data Link Layer Control Methods for Performance Improvement of TCP over an Ad Hoc Network

Takayuki YAMAMOTO[†], ***Student Member***, **Taichi YUKI**[†], ***Nonmember***, **Masashi SUGANO**[††], **Masayuki MURATA**[†], ***and*** **Takaaki HATAUCHI**[†††], ***Members***

SUMMARY

In order to increase TCP performance in an ad hoc network, support from the lower layers of the transport layer is required. Although retransmission by the data link layer is generally required for errors on a wireless channel in ad hoc networks, duplicate packets due to the disappearance of a receipt check can be generated at the same time and increase network vain load. In the high load networks, packets tend to collide, and duplicate packets are often generated. Once loads begin to become high, the loads will increase further and further. Although TCP controls congestion, TCP tends to experience such a situation, because TCP cannot respond to the increase of the load by duplicate packets. In this paper we propose a technique of performing retransmission of the data link layer efficiently according to the load of the nodes. We have found that throughput can be improved by up to 16% with this technique by means of simulation experiments.

key words: *Ad hoc networks, TCP (Transmission Control Protocol), Data link layer, Retransmission, Throughput, Simulation*

1. Introduction

In recent years, the application area of ad hoc networks has been expanding, and the interconnection with wired networks and the demand for the same service offered in wired networks are increasing. In wired networks, since TCP (Transmission Control Protocol) is generally used as a transport layer protocol, it is used for communication also in ad hoc networks. Generally, a wireless channel has a more unstable transmission quality than a wired circuit, and since packet loss occurs frequently, also in a wireless network by the cellular communication system, the technique for transmitting TCP efficiently has been a problem. Moreover, since an ad hoc network becomes a multi-stage composition of a wireless channel and movement of terminals is generated, cutting of a connection and change of a route will tend to take place, and the performance of TCP will deteriorate remarkably. Therefore, also in an ad hoc network, it is important to establish a technique for transmitting TCP efficiently.

Manuscript received December 24, 2004.

[†]The authors are with the Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565-0871 Japan.

[††]The author is with the Faculty of Comprehensive Rehabilitation, Osaka Prefecture College of Nursing, Habikino-shi, 583-8555 Japan.

[†††]The author is with Fuji Electric Advanced Technology Co.,Ltd., Hino-shi, 191-8502 Japan.

Many studies to the improvement of TCP performance have been dedicated over ad hoc networks [1-8]. Much of this research has focused on TCP performance degradation caused by terminal movement. For example, [5] and [6] produce techniques that distinguish route failure and congestion, and decide whether to control congestion or not. Therefore, they can cope with situations correctly. In [7] and [8], new transport layer protocols that are suitable for ad hoc networks are introduced. However, in this paper, we do not modify TCP because we regard the seamless communication between wired networks and ad hoc networks as the most important subject.

In data link layer, retransmission is generally required for errors on a wireless channel in ad hoc networks. However, if a receipt check is lost by an error or a collision, duplicate packets will occur at the same time and increase the vain load of the network. In a high load network, packets tend to collide and duplicate packets are often generated. Namely, once the load begins to become high, the load will increase further and further. Although TCP controls congestion, TCP tends to experience such a situation, because TCP raises the window size in order to obtain as good a throughput as possible and cannot correspond to the increase of the load by duplicate packets peculiar to an ad hoc network. Therefore, we propose a technique of performing retransmission of the data link layer efficiently according to the load of nodes. If the load of a network is high, since the collision probability of a packet is high, the interval in which to retransmit a packet is extended. Conversely, if the load of a network is low, the response time of TCP will be shorter by retransmitting quickly. To evaluate this technique, we applied it to an ad hoc network that uses table-driven routing with fixed terminals [9] In a fixed ad hoc network, packet losses are caused mainly by packet collisions rather than by node mobility. In this way, we clearly see the effect of this technique.

This paper is organized as follows. In Section 2, we begin by describing the target ad hoc network system. In Section 3, we investigate the problem of collisions of packets and the high incidence of duplicate packets due to the high load of the network and propose a technique to control retransmission. Then, we evaluate this technique by means of simulation in Section 4. Finally, we

conclude this paper and outline future research topics in Section 5.

2. System Description of Target Ad Hoc Network

In this section, we introduce an ad hoc network system this research targets. The Flexible Radio Network (FRN) is a multi-hop wireless network system. We omit the details of the protocol of FRN and describe only the portion relevant to this paper.

In the FRN, the wireless channel is divided into fixed-length time slots. When a packet is to be sent, the node wanting to send the packet does a carrier sense at the start of the time slot and this carrier sense prevents the packet from colliding with another. In addition, acknowledgement between neighboring nodes is done using a packet which is also used for forwarding from one node to the next. Every neighboring node in a wireless network can receive packets from a node even when it is not the packet source/destination. We call such a packet a *relay echo* in the FRN. The final destination node of a packet does not forward the packet, instead sending a relay echo, and so it sends a FRN ACK packet to the previous node. Forwarding of a packet and sending of a FRN ACK are done in the time slot immediately after the slot in which the node receives the packet.

When the transmission of a packet fails because of a link failure or packet collision, the node resends the packet after waiting for a random number of time slots to prevent another packet collision. Although this random number of time slots is generally within a range of three to five slots, the most desirable number of slots is undetermined. We therefore examined the interval of random time slots, setting it as 3-5 slots (the conventional number) as the shortest interval, 3-9 slots, 3-13 slots, $\cdots$,and 3-25 slots as the longest interval.

3. Technique to Control Packet Retransmission

3.1 Packet Collisions in High Load Situations

When TCP is used over an ad hoc network, TCP raises window size in order to obtain a better throughput. By this mechanism, the network load becomes high automatically and collisions of packets occur frequently. Moreover, duplicate packets due to the disappearance of receipt checks increase load and may degrade performance. The process in which the duplicate packets generate is shown in Figure 1. Figures 1(a) and 1(b) show that a packet is forwarded to a destination node. However, if a relay echo of the packet collides with another packet (Figure 1(c)), the node will not know that the packet could be forwarded correctly and will forward the packet to another node again (Figure 1(e)).

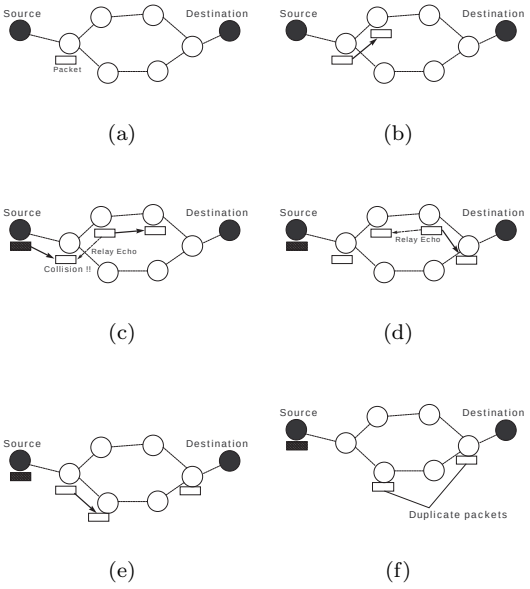


Fig. 1 An example of a process of duplicate packets generation

In addition, in Figure 1(f), duplicate packets exist in the network. Therefore, once the load increases, it will continue to increase and the network will fail eventually. Although TCP controls congestion, TCP tends to experience such situations, because TCP raises window size in order to obtain as good a throughput as possible and cannot correspond to the increase of the load by duplicate packets peculiar to an ad hoc network.

We then noted that retransmission by the data link layer overlaps retransmission of TCP and considered suppressing unnecessary retransmission by the data link layer. When the load of the network is light and when the error rate of a wireless channel is low, the probability that a packet will be transmitted correctly is high. Therefore, we felt that the necessity for retransmission by the data link layer is low in that situation and did not want to retransmit by the data link layer. However, it turns out that such a method is difficult. The state where the error rate is high or the high state of the network load have a high probability that a relay echo will not come by a collision or a loss, etc., of a packet, and the receipt check of transmission becomes difficult. Retransmitting in such a state causes the generation of duplicate packets and leads to further load increase. Therefore, it is difficult to control whether it retransmits by the data link layer or not by means of the error rate or network load.

3.2 Proposed Technique

Here, we propose a technique for improving the throughput of TCP by controlling the interval of retransmission according to the load of a node and retransmitting packets efficiently. When the load of a

Table 1 An example of transmitting history

Sequence Number	3	4	4	···
Destination	Node 5	Node 5	Node 2	···
Time	1003.03	1004.10	1004.60	···
OK or NG	○	×	○	···

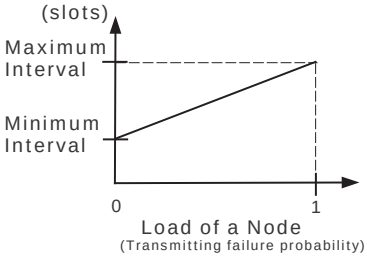


Fig. 2 Changing the retransmission interval by load

node is high, in order to avoid a collision, dispersion of the interval of retransmission is enlarged. Conversely, when the load of a node is low, dispersion of a retransmitting interval is made small. By this technique, when load is high, the number of collisions decreases and a duplicate packet generation is suppressed. When load is low, a packet is retransmitted quickly, the response time of TCP becomes short, and the throughput of TCP improves.

We shall now describe the detailed contents of this proposal. We regard the transmitting failure probability of a node as the load of the node circumference, because a possibility that a packet will collide and transmission will go wrong is high when load is high. Each node preserves the transmitting history of the packet sent from itself, and saves whether the transmission succeeded or not, as shown in Table 1. A certain number of history is held during a fixed time. Each node gets transmitting failure probability from this history and regards it as the load of the node circumference. Next, each node controls the dispersion of a retransmission interval from its load. The dispersion of the retransmission interval is extended in proportion to the load, as shown in Figure 2. A horizontal axis is the load drawn from the transmitting history and a vertical axis serves as an interval corresponding to it. A node selects the minimum interval when the node judges that there is no load. The interval is extended according to the increase of the load and reaches the maximum at last.

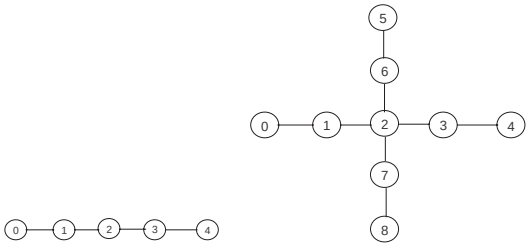
Since the time to stay in each relay node will be changed by the load when this technique is used, it becomes difficult to set up a maximum lifetime by the source node. Therefore, we decided to discard a packet on the basis of the retransmission count that the packet experienced, because a packet with many retransmissions has a high possibility of being a duplicate packet (as we explained in Figure 1) and has a high possibility of not reaching a destination node. Specifically, every packet records the number of retransmissions in its header and will be discarded when the value exceeds a threshold that is set up by the source node. We call this threshold the "maximum count of retransmission" and will evaluate how this value should be set up in the following section.

We describe the difference between our proposal and backoff mechanism of CSMA/CD (Carrier Sense Multiple Access with Collision Detection). In CSMA/CD, if a transmission of a certain frame goes wrong, the retransmission will be delayed exponentially. In that a retransmission interval is changed according to load, our proposal and CSMA/CD are similar. However in CSMA/CD, a node selects the shortest retransmission interval with the following frame, if it succeeds in transmitting one frame. This shows that load is not reflected correctly in CSMA/CD. We think that the high load situation continues for a while. The width of a retransmission interval should reflect the load at that time, and it should not be initialized if a transmission of one packet is successful. If a retransmission interval is initialized for every successful transmission, collisions will occur frequently and the performance will deteriorate, especially in a high load situation. Therefore, our proposal that refers to load and controls retransmissions is effective.

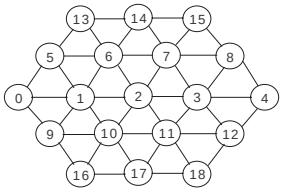
4. Simulation and Evaluation

4.1 Simulation Model

We evaluated our proposed technique through simulations using ns-2 [11] with its radio propagation model extended by the CMU Monarch Project [12]. We used the IEEE 802.11 multicast transmission mode for all packet transmissions with a slight modification to simulate the FRN time slots. In all simulations, the time slots were synchronized at all nodes. This mode is a single-hop multicast that does not produce the channel reservation mechanism that is produced by RTS/CTS of the IEEE 802.11 unicast mode. The radio transmission range was 250 m and each node's buffer capacity was large enough to inhibit buffer overflow in our simulations. Each node exchanged its network configuration table at intervals sufficiently long to not affect the system performance. We used throughput as a measure of performance. The throughput was defined as the average number of acknowledged data packets sent from every node per time slot. That is, we measured the total network performance. Furthermore, we used the network topology of Figure 3. A circle and a number in the circle mean a node and its address. A line connecting two nodes means that they can communicate directly.

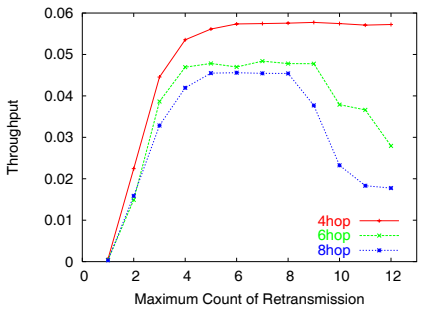


(a) Chain topology (b) Cross-chain topology

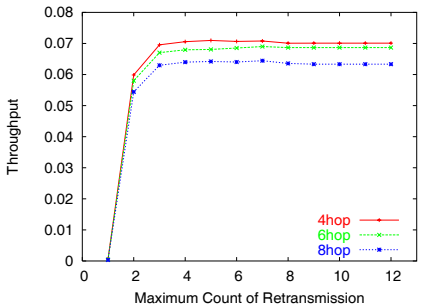


(c) Mesh topology

Fig. 3 Network topologies



(a) Interval of retransmission: 3-5 time slots



(b) Interval of retransmission: 3-17 time slots

Fig. 4 Change of the throughput according to a maximum count of retransmission without the error of a wireless channel in chain topologies

4.2 Maximum Count of Retransmission

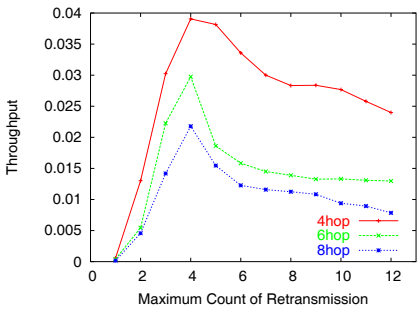
First, we evaluated the maximum count of retransmission. We observed change of the throughput when changing the maximum count of retransmission by using the chain topologies of 4 hop, 6 hop, and 8 hop. By using these topologies, we understood the influence that the hop number has on the maximum count of retransmission. We generated the random error of a wireless channel as 0% and 5%.

We show change of the throughput by the count of maximum retransmission when not generating the random error of a wireless channel in Figure 4. Figure 4(a) shows change of the throughput when the retransmission interval is set as 3-5 time slots. In this figure, if the maximum count of retransmission is set as a large value, the throughput of the connection of 6 hop and 8 hop will fall. This is because packets will pile up on the middle nodes and raise the load of a network, if the maximum count of retransmission is enlarged through a connection with a large hop number when a retransmission interval is short and the possibility of collisions is high. We show the throughput when setting a retransmission interval as 3-17 time slots in Figure 4(b) and the connection of every hop number has the same tendency. This is because we set a large retransmission interval and collisions of packets do not occur frequently.

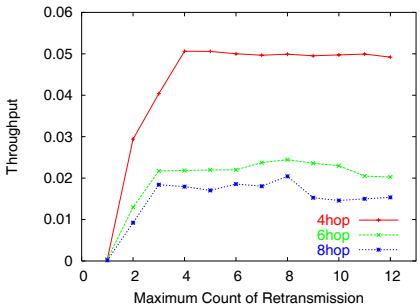
Figure 5 shows the result when setting the rate of random error of a wireless channel at 5%. It turns out that packet stays occur frequently and the throughput becomes low in a smaller maximum count of retransmission (Figure 5(a)). Moreover, from Figure 5(b), although packet collisions decrease since the retransmission interval is lengthened, when the error rate of a wireless channel is high, the possibility that retransmission for which it waited so long will go wrong is high. For this reason, piling up of packets occurs from a small maximum count of retransmission and change of throughput by the maximum count of retransmission is no longer seen.

Although we considered it better for the connection with a large hop number to make a large maximum count of retransmission, it is shown that this is not correct. The connection with a large hop number tends to generate collisions of packets, and also tends to generate packet pile ups. This result showed that it was better to make the maximum count of retransmission small to some extent and to prevent packet stays. Moreover, when the rate of an error is made high in the connection of a long hop and a retransmission interval is set up short, there is a place where the throughput becomes good. This shows that it is meaningless to delay retransmissions, because piling up of packets may occur, and this may lead to the fall of a throughput when the error rate of a wireless channel is high.

Next, we set the maximum count of retransmis-

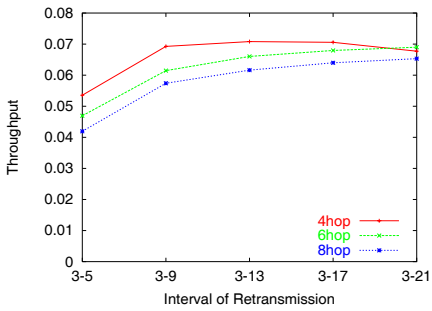


(a) Interval of retransmission: 3-5 time slots

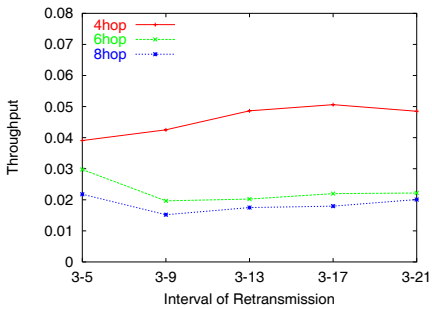


(b) Interval of retransmission: 3-17 time slots

Fig. 5 Change of the throughput according to a maximum count of retransmission with the 5% random error of a wireless channel in chain topologies



(a) The error rates of a wireless channel: 0%



(b) The error rates of a wireless channel: 5%

Fig. 6 Change of the throughput according to the retransmission interval

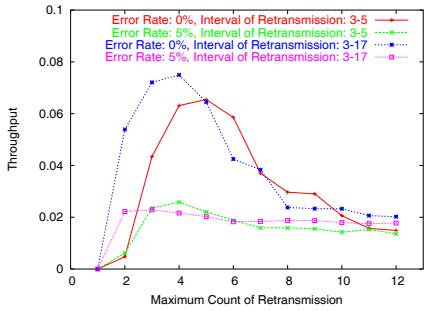
sion at 4 and show the throughput when changing the dispersion of a retransmission interval in Figure 6. Figure 6(a) is the result when not generating the error of a wireless channel, and we can see the fall of the throughput when the retransmission interval is lengthened with 3-21 time slots. If a retransmission interval is extended too much, end-to-end transmission will become slow and the throughput will fall. Figure 6(b) is the result when generating the random error of a 5% wireless channel. In the topology of the large hop, the retransmission interval of 3-5 time slots obtained the best throughput. This is because the waiting time is long when error of a wireless channel and piling up of a packet have occurred if a retransmission interval is made in 3-9 time slots.

We observed change of the throughput by the maximum count of retransmission for various topologies. The results are shown in Figure 7. The tendency of every topology is almost the same and 3, 4, or 5 times of maximum retransmission count become the best throughputs, as before. Therefore, we set the maximum count of retransmission at 4 when we evaluate our proposal from now on.

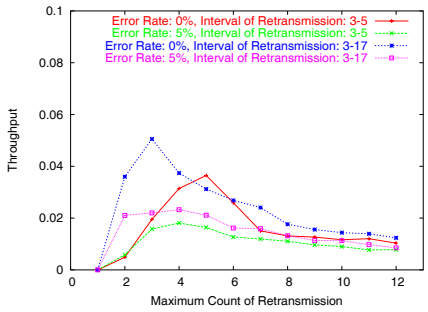
4.3 Evaluation of Proposed Technique

First, we simulate the technique of changing the dispersion of a retransmission interval dynamically according to the load of a node by using chain topologies of 4 hop and 8 hop. We set the error of the wireless channel at 0%, set the maximum count of retransmission at 4, and set the number of transmitting history at 10 for the present. The results are shown in Figure 8. The line of the graph has the the same dispersion of the shortest retransmission interval when judging the load to be nothing. The dispersion of the longest retransmission interval becomes large, so that it travels on a horizontal axis to the right. That is, if the graph of "Minimum Interval:3-5" becomes the retransmission interval of 3-5 when the load is 0, and the "Maximum Interval" of the horizontal axis is enlarged, the degree that extends the dispersion of a retransmission interval by load will become large.

In the case of a 4-hop chain topology (Figure 8(a)), the maximum throughput when nodes control the dispersion of a retransmission interval dynamically with the load (shortest dispersion: 3-5 time slots, longest dispersion: 3-13 time slots) improves about 6% over

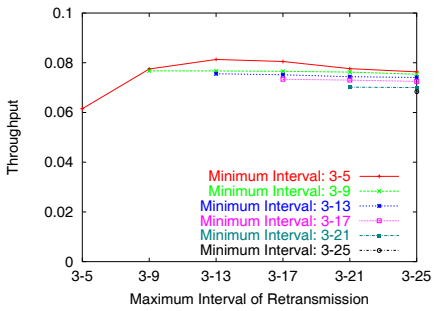


(a) 2 connections of 8 hop in cross-chain

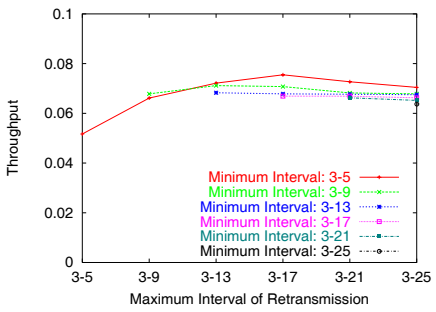


(b) mesh topology

Fig. 7 Change of the throughput according to a maximum count of retransmission in cross-chain topologies



(a) 4-hop chain topology



(b) 8-hop chain topology

Fig. 8 Change of the throughput when changing a retransmission interval dynamically

the maximum throughput when nodes do not change the dispersion (fixed retransmission interval: 3-9 time slots). We evaluate by changing the transmitting history number (Figure 9). The rate of improvement becomes small when we set a small number for the transmitting history, such as 2, 4, and 6. When we set a large number such as 12, the rate of improvement becomes small similarly. This shows that the load cannot be judged correctly with only a few histories. Conversely, when a node has many histories, the node cannot respond to the load change. Therefore, we set the number of the transmitting history at 10 from now on.

In the case of an 8-hop chain topology (Figure 8(b)), the throughput when setting a retransmission interval to the shortest 3-5 time slots and longest 3-17 time slots improved about 10% over a throughput at the fixed retransmission-interval time of 3-13 time slots. The rate of improvement of the 8 hop is higher than that of the 4 hop. We think that this is because there is a place of high load and low load since the number of the hop is long. Therefore, we look at the effect of the technique by the various topologies.

We evaluate our proposal in a 4-hop cross-chain topology, 8-hop cross-chain topology, and 19node mesh topology. The results are shown in Figures 10 and 11. In the 4-hop cross-chain topology, the tendency changes

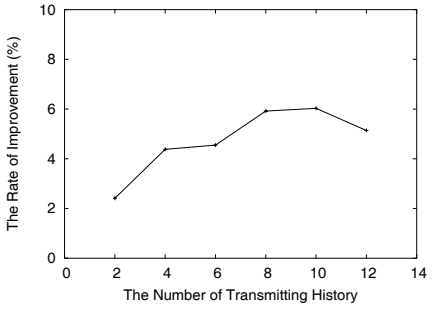
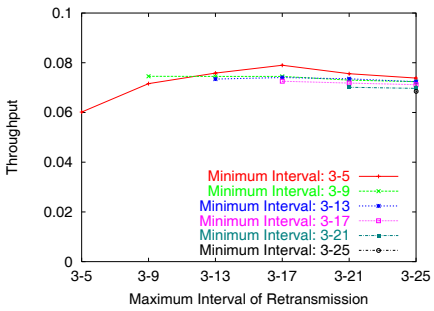
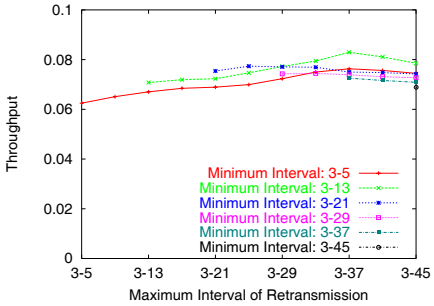


Fig. 9 The rate of improvement in each number of transmitting history

little from the previous chain topology. This is because the topology is quite narrow and there is little load change within the topology. In this case, the throughput when setting a retransmission interval to the shortest 3-5 time slots and longest 3-17 time slots improved about 6% over the throughput at the time of the fixed retransmission-interval at the 3-9 time slots. In the 8-hop cross-chain topology and in mesh topology, the rate of improvement is 10% and 16%, respectively. These topologies have many nodes and nodes from which the loads differ from each other occur easily. Therefore, the rate of improvement becomes high in these cases.



(a) 4-hop cross-chain topology



(b) 8-hop cross-chain topology

Fig. 10 Change of the throughput when changing a retransmission interval dynamically in cross-chain topologies

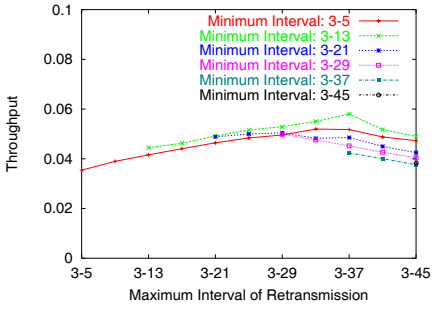
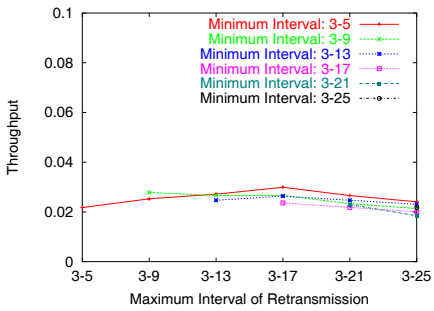
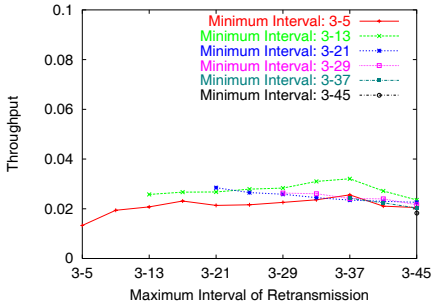


Fig. 11 Change of the throughput when changing a retransmission interval dynamically in mesh topology

Finally, we observe the effect of our proposal when generating the random error of a wireless channel. We made the probability of a random error 5%, and chose an 8-hop chain topology and mesh topology in brief. The results are shown in Figure 12. Although the throughput falls on the whole, the two results are almost same as those when we did not generate the random error of a wireless channel. In addition, in the case of an 8-hop chain topology, the rate of improvement of a throughput is 7.5%. In the case of mesh topology, it is 12.7%. This shows that the proposal is effective also in the high state of an error, although the rate of improvement falls a little compared with the case where an error does not occur.



(a) 8-hop chain topology



(b) Mesh topology

Fig. 12 Change of the throughput when changing a retransmission interval dynamically with 5% random error of a wireless channel

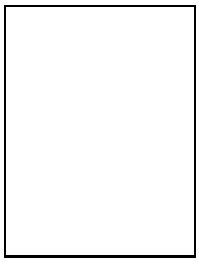
5. Conclusion

In this paper, we focused on the increased load and packet collisions in TCP, and the fact that packet collisions generate duplicate packets. Therefore, we proposed a technique that adjusts the interval of retransmission by the load of a node, and showed through simulation that this technique can retransmit efficiently according to the load of the node. In the simulation, the technique improved the TCP performance by up to 16%. We also showed that the technique is effective in the case when errors occur.

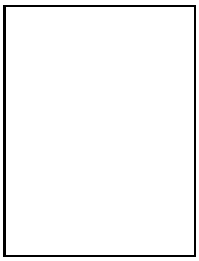
By these data link layer techniques, TCP's performance has been improved. However, we know that support from the lower layer of TCP is not only from the data link layer but also from the network layer. Therefore, we will develop a routing protocol suitable for using TCP in ad hoc networks. For example, as it is possible that many routes exist in a network where many nodes exists, we can perform routing so that load may be distributed, packet collisions reduced, and TCP performance improved.

References

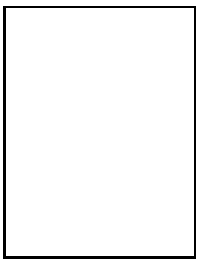
[1] D. Kim, C.K. Toh, and Y. Choi, "TCP-BuS: improving TCP performance in wireless ad hoc networks," Proceedings of IEEE ICC 2000, pp.1707–1713, June 2000.
[2] G. Holland and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks part I: Problem discussion and analysis of results," Proceedings of ACM/IEEE MOBICOM'99, pp.219–230, Aug. 1999.
[3] S. Bansal, R. Shorey, S. Chugh, A. Goel, K. Kumar, and A. Misra, "The capacity of multi-hop wireless networks with TCP regulated traffic," Proceedings of IEEE GLOBECOM 2002, Nov. 2002.
[4] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP throughput and loss," Proceedings of IEEE INFOCOM 2003, March 2003.
[5] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A feedback-based scheme for improving TCP performance in ad hoc wireless neworks," IEEE Personal Communications Magazine, vol.8, no.1, pp.34–39, Feb. 2001.
[6] J. Liu and S. Singh, "ATCP: TCP for mobile ad hoc networks," IEEE Journal of Selected Areas in Communications, vol.19, no.7, pp.1300–1315, July 2001.
[7] Z. Fu, B. Greenstein, X. Meng, and S. Lu, "Design and implementation of a TCP-friendly transport protocol for ad hoc wireless networks," Proceedings of IEEE ICNP'02, pp.216–225, Nov. 2002.
[8] K. Sundaresan, V. Anantharaman, H.Y. Hsieh, and R. Sivakumar, "ATP: a reliable transport protocol for ad hoc networks," Proceedings of ACM MobiHoc 2003, pp.64–75, June 2003.
[9] "Flexible Radio Network, Fuji Electric Co. Ltd." available at `http://www.fujielectric.co.jp/denki/p26/ecop_contents2.html`.
[10] C.E. Perkins, Ad Hoc Networking, Addison-Wesley, 2001.
[11] "The Network Simulator - ns-2." available at `http://www.isi.edu/nsnam/ns/`.
[12] "The CMU monarch project." available at `http://www.monarch.cs.cmu.edu/`.



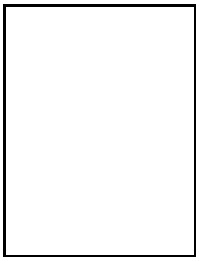
Takayuki YAMAMOTO received the M.E. degree in Information and Computer Science from Osaka University, Japan, in 2002. He is now a doctoral student at the Graduate School of Information Science and Technology, Osaka University. His research interests include wireless ad hoc networks and their performance evaluation and simulation. He is a student member of IEICE.



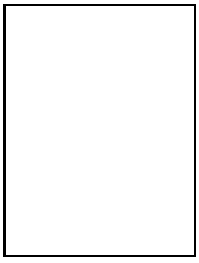
Taichi YUKI received the B.E. degree in Information and Computer Science from Osaka University, Japan, in 2002. He is now a student in the master's course at the Graduate School of Information Science and Technology, Osaka University. His research interests include fusion by wireless ad hoc networks and wired networks, their performance evaluation and simulation.



Masashi SUGANO received the B.E., M.E., and D.E. degrees in Information and Computer Sciences from Osaka University, Japan, in 1986, 1988, and 1993, respectively. In 1998, he joined Mita Industrial Co., Ltd. (currently, Kyocera Mita Corporation) as Researcher. From 1996 to 2003, he was an Associate Professor in Osaka Prefecture College of Health Sciences. He moved to Faculty of Comprehensive Rehabilitation, Osaka Prefecture College of Nursing in April 2003. His research interests include design an performance evaluation of computer communication networks, network reliability, and wireless network systems. He is a member of IEEE, ACM, IEICE and IPSJ.



Masayuki MURATA received the M.E. and D.E. degrees in Information and Computer Sciences from Osaka University, Japan, in 1984 and 1988, respectively. In April 1984, he joined Tokyo Research Laboratory, IBM Japan, as a Researcher. From September 1987 to January 1989, he was an Assistant Professor with Computation Center, Osaka University. In February 1989, he moved to the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University. From 1992 to 1999, he was an Associate Professor in the Graduate School of Engineering Science, Osaka University, and from April 1999, he has been a Professor of Osaka University. He moved to Graduate School of Information Science and Technology, Osaka University in April 2004. He has more than three hundred papers of international and domestic journals and conferences. His research interests include computer communication networks, performance modeling and evaluation. He is a member of IEEE, ACM, The Internet Society, IEICE and IPSJ.



Takaaki HATAUCHI was born in Hiroshima, Japan, in 1959. He received the B.E. degree from Kinki University in 1982. He joined Fuji Electric. His current research interests are communication protocols for wireless system.