

# Improvement of TCP Throughput by Combination of Data and ACK Packets in Ad Hoc Networks

Taichi YUKI<sup>†</sup>, *Nonmember*, Takayuki YAMAMOTO<sup>†</sup>, *Student Member*, Masashi SUGANO<sup>††</sup>, Masayuki MURATA<sup>†</sup>, Hideo MIYAHARA<sup>†</sup>, and Takaaki HATAUCHI<sup>†††</sup>, *Members*

**SUMMARY** Since a radio channel is shared among terminals in an ad hoc network, packet collisions are frequent. When transmitting packets using TCP, data and ACK packets are transmitted in opposite directions on the same radio channel. Therefore, frequent collisions are unavoidable, and this seriously degrades throughput. To reduce the likelihood of packet collisions when an intermediate node transmits both data and ACK packets, these two types of packet can be combined and transmitted at the same time to increase the efficiency of radio channel utilization.

In this paper, we propose a new technique to improve TCP performance by combining data and ACK packets. Our proposed technique is applicable to generic ad hoc networks easily. By means of a simulation using networks with various topologies, we have found that throughput can be improved by up to 60% by applying our proposed technique.

**key words:** *ad hoc network, TCP, collision, throughput, simulation*

## 1. Introduction

In recent years, many ways of applying wireless ad hoc networks have been developed. When an ad hoc network provides the same services as a wired network, TCP [1] should be used as the transport layer protocol because it is the de facto standard for wired networks. An ad hoc network is multi-hop network composed of wireless channels, though, and the transmission quality of a wireless channel is less stable than that of a wired channel. Therefore, packet loss occurs frequently in an ad hoc network, and the consequent connection failure can severely degrade TCP performance. Several groups have studied TCP performance over ad hoc networks [2-9]. Much of this research has focused on TCP performance degradation caused by terminal movement. For example, [2] developed the explicit link failure notification (ELFN) technique. ELFN reduces the effect of the decrease of the TCP window size when a link failure occurs in the middle of a route. In this technique, a node freezes the TCP mechanism when a link

failure occurs, thus preventing the TCP from making the window size excessively small, and so the TCP performance is improved. Although [2] focused on node mobility, we have examined the performance degradation caused by short-duration link failure in an ad hoc network where the terminal is stationary. We have applied ELFN and developed a technique that improves the performance of such a network [10]. [6] and [7] produce techniques that distinguish route failure and congestion, and decide whether control congestion or not carry out. So they can cope with situations correctly. In [8] and [9], new transport layer protocols are introduced that are suitable for ad hoc networks. But in this paper, we do not change TCP because we regard the seamless communication between wired networks and ad hoc networks as the most important subject.

However, we found that simply avoiding link failure was not sufficient to improve TCP performance. Packet collisions occur because TCP is based on bidirectional communication. When TCP is used as the transport layer protocol, a TCP sender sends data packets and a TCP receiver receiving packets sends ACK packets to a sender for acknowledgement. In ad hoc networks, since nodes cannot distinguish between data packets and ACK packets, collisions often occur over wireless connections. As packets travel over multi-hop links, they often collide. The IEEE 802.11 standard provides for channel reservation based on an RTS/CTS control message, but this causes another problem. Neighboring nodes that can receive radio-wave signals must be silent until the channel is released, especially in a high-density network topology. Many nodes cannot send packets they want to send, and eventually packet losses occur and performance deteriorates [11].

In this paper, we propose a technique to improve TCP performance in an ad hoc network that focuses on the bidirectional characteristic of TCP. In this technique, if a data packet and an ACK packet meet in an intermediate node, they will be collectively transmitted in an opposite direction simultaneously. In this way, packet collisions can be avoided and effective use of a wireless channel achieved.

To evaluate this technique, we applied it to an ad hoc network which used table-driven routing with fixed terminals. In a fixed ad hoc network, packet losses are

Manuscript received January 1, 2002.

Manuscript revised January 1, 2003.

Final manuscript received January 1, 2004.

<sup>†</sup>The author is with Graduate School of Information Science and Technology, Osaka University, Toyonaka-shi, 560-8531, Japan.

<sup>††</sup>The author is with Osaka Prefecture College of Nursing, Habikino-shi, 583-8555, Japan.

<sup>†††</sup>The author is with Energy and Electrical Systems Company, Fuji Electric Co., Ltd., Hino-shi, 191-8502, Japan.

caused mainly by packet collisions rather than by node mobility. In this way, we could clearly see the effect of this technique. Flexible Radio Network (FRN) is a commercially available product based on an ad hoc network system, and it is driven by a routing table with fixed nodes [12]. We therefore used FRN to evaluate our proposed technique. Through a simulation using networks with various topologies, we found that throughput could be improved by up to 60%, or at least 10% in a very high load situation, by applying our proposed technique.

This paper is organized as follows. Sec. 2 describes the FRN system and Sec. 3 describes our proposed technique. We evaluate the technique in Sec. 4, and discuss the applicability of our proposed technique to other existing ad hoc networks. We conclude the paper in Sec. 5.

## 2. System Description of FRN

FRN is a commercial product based on ad hoc network technology. A network can be built only by suitably placing terminal nodes, and the network can be extended only by adding nodes as needed. FRN has been used, for example, for electric energy control in factories and ticket management systems in skiing areas. Services usually supplied over a wired network are now being requested through an FRN. In such a system, all nodes are controlled by a protocol which can efficiently adapt to node failures and changes to the network configuration.

An FRN's routing system is table-driven, like a DSDV (destination-sequenced distance vector) type [13] with periodic communication. Routing protocols of an on-demand type, such as AODV (ad-hoc on-demand distance vector) [13], are suitable for networks whose nodes move rapidly. However, FRN nodes are basically stationary, they can know their neighboring nodes, and manage through a routing table. Later on, we outline the FRN routing method and the FRN data-link protocol. The details of these are given in [14].

### 2.1 Data-link Protocol

In an FRN, the radio 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 an FRN. The final destination node of a packet does not forward the packet, instead sending a relay echo, and so it sends an FRN ACK

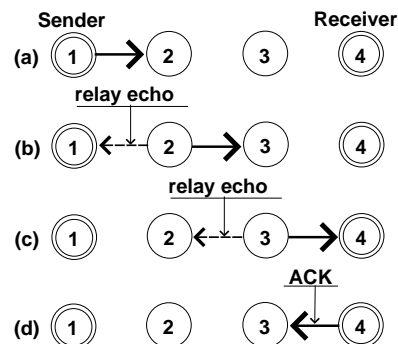


Fig. 1 Relay echo mechanism

packet to the previous node. Forwarding of a packet and sending of an FRN ACK are done in the time slot immediately after the slot in which the node receives the packet. Fig. 1 shows the relay echo mechanism.

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, ... and 3-25 slots as the longest interval.

The maximum lifetime - the maximum time that a packet is allowed to exist within a network - is defined by slot for all packets and set at the source node. This lifetime is decreased by one for every time slot even if the packet remains in a buffer. When the value reaches zero, the packet is discarded. In the original FRN system, the value of this parameter was defined to be long enough for a network scale. If the value is too small, packets cannot reach their destination; if the value is too big, unneeded packets remain in the network for a long time. Therefore, this value is very important. In this paper, we tentatively set it as 32 slots.

### 2.2 Routing Protocol

In the FRN, each node manages network information in a *network configuration table*. The network configuration table contains the route information from the node to each destination node. The route information consists of a list of the neighboring nodes' addresses on the routes to a destination node and the hop count of each route. This network configuration table is created through periodic exchanges of control packets that contain information regarding the shortest route.

Every node maintains multiple sets of route information for each destination node, and selects one when sending packets to that node. This selection method

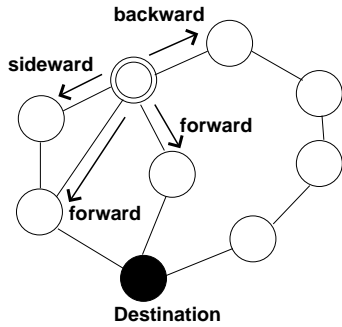


Fig. 2 Classification of adjacent nodes

is as follows. For each destination, routes are classified into three groups according to their hop count (Fig. 2).

- Forward route: The route(s) having the lowest hop count to the destination.
- Sideward route: The route(s) whose hop count to the destination is equal to the shortest hop count plus one.
- Backward route: The route(s) whose hop count to the destination is equal to the shortest hop count plus two or more.

The transmitting-priority order with respect to which nodes to send to is forward route, sideward route, and backward route. If every transmission to a node on a forward route fails, transmission to a sideward node is attempted. If transmission along all possible sideward routes also fails, the node transmits to a backward node.

### 3. The Proposed Technique

We explained the problems that arise in ad hoc networks in Sec. 1. In this paper, we focus on the problem of packet collisions caused by bidirectional TCP communication.

Here we explain our proposed technique to alleviate the problem of packet collisions. ACK packets are very small, containing only TCP header information. Transmitting such an ACK packet using a time slot as big as that used for a data packet wastes radio channel capacity. Therefore, we have considered ways to transmit a combined data and ACK packet by exploiting a characteristic of a wireless channel: that all nodes within the range of an electric wave used to transmit a packet can know the packet contents.

To put it more concretely, every node needs to have two queues. Data packets and ACK packets are saved in their respective queues. When a packet is in both of queue, each destination is determined, and the combined packet is transmitted in a form where an ACK packet is added to a data packet. If each node has only one queue, the combined data and ACK packet can be

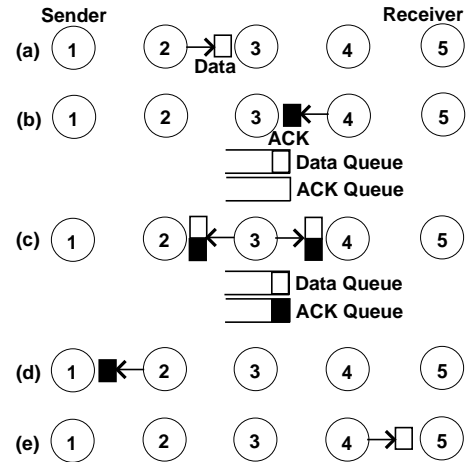


Fig. 3 A process where the proposed technique is used

saved in the queue. However, the combined packet will be erased when one relay echo or ACK (not of TCP but of FRN's data-link layer) arrived in this case, and the function of the relay echo will be destroyed.

Fig. 3 shows a process where our proposed technique is used. If a node does not have packets in each queue, the node behaves as before. If a node has packets in each queue, the node combines a data packet and an ACK packet from the top of each queue and sends the combined packet to two destinations (Fig. 3(c)). If a node receives a combined packet, it then determines the two destinations of the combined packet, and when that node is one of the destinations, it receives the portion of the packet addressed to itself. If the node is not one of the destinations, it discards the packet. Here, we must be careful regarding the time slot that the next-hop nodes use to forward the packets. If the nodes forward the packet in the next time slot, as before, packet collision invariably occurs and the node that sent the combined packet cannot receive each relay echo. To avoid this, we set up the time slot so that a packet is less likely to collide with another when nodes receive the combined packet. Fig. 3(d) and 3(e) show this. Node 4 received the data-packet portion of the combined packet, so it postpones retransmitting the ACK-packet portion for one time slot to prevent a collision. Node 3 can thus receive each relay echo from nodes 2 and 4.

This technique enables more efficient use of the radio channel and reduces the chance of packets colliding when a node has both data and ACK packets. When TCP is used as the transport layer protocol, there are bidirectional streams in the network, so intermediate nodes often possess both data and ACK packets. Thus, our technique should significantly improve TCP performance.

We evaluated the effect of the delayed ACK option of TCP [1]. The delayed ACK option is aimed at

effective use of a wireless channel by sending collected ACK segments. When a destination node receives a data packet, the node delays the return of an ACK packet for a fixed time. All of a node's accumulated ACK packets can be transmitted as one ACK packet if another data packet is receivable within this time. If the number of ACK packets can be lowered by using this option, improved TCP throughput in an FRN can be realized. Therefore, we next evaluated whether our proposed technique is also effective when used simultaneously with this option.

## 4. Evaluation

### 4.1 Simulation Environment

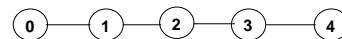
We evaluated our proposed technique through simulations using ns-2 [15] with its radio propagation model extended by the CMU Monarch Project [16]. We used the IEEE 802.11 multicast transmission mode for all packet transmissions with a slight modification to simulate the FRN time slots. In all simulations, 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 set the maximum lifetime as 32 slots in all simulations. This time slots is moderate length for topologies of all simulations.

We used the network topologies shown in Fig. 4. 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. Although these topologies are very simple, we can use them to identify basic tendencies of our proposed technique and apply the results to the general FRN network. In all simulations, we used TCP Reno as the transport layer protocol. We also evaluated the case where the delayed ACK option was used. We used the Fig. 4(a) topology to evaluate the technique in a pure bidirectional connection. In Fig. 4(b), there is a crossing of connections and we evaluated the technique in such a case. Fig. 4(c) shows a mesh topology, a more complicated topology with more random connections.

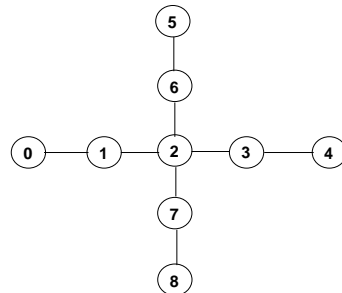
We used throughput as a measure of performance. The throughput was defined as the average number of acknowledged data packets sent from every node per time slot. That is, we measured the total network performance.

### 4.2 Results and Discussions

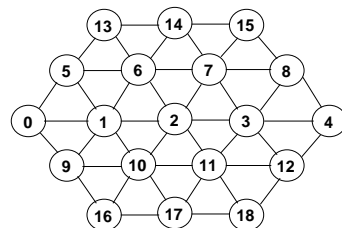
First, we evaluated throughput with the simplest topol-



(a) Chain Topology



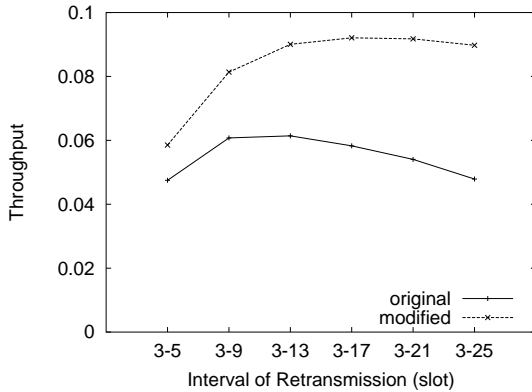
(b) Cross-Chain Topology



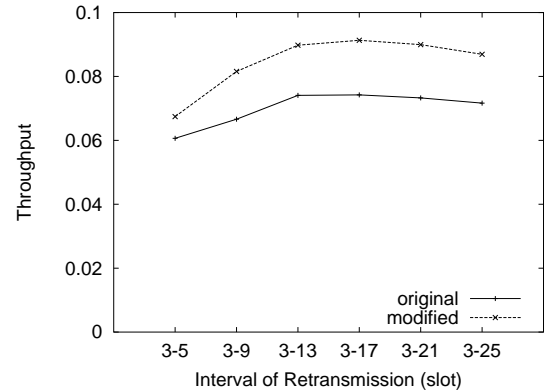
(c) Mesh Topology

**Fig. 4** Network topologies

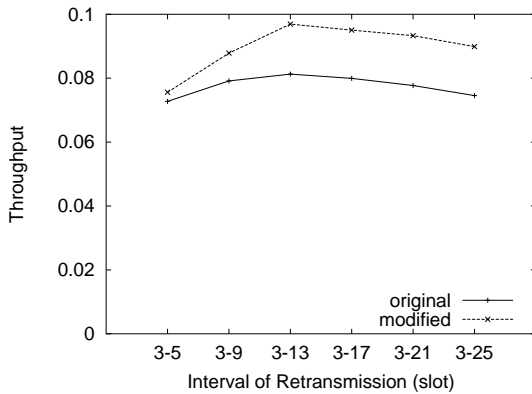
ogy (Fig 4(a)). In this network, node 0 was a TCP sender and node 4 was a receiver, so this connection looked like a chain of 4 hops. The results are shown in Fig 5. As mentioned (Sec. 2), we changed the retransmission interval from 3-5 time slots to 3-25 time slots and measured the throughput at each interval. Fig. 5(a) and Fig. 5(b) show, respectively, the results without and with the delayed ACK option. A similarity in the two figures is that throughput is low with a short retransmission interval, improves as the interval becomes longer, and eventually deteriorates again at the longest intervals. This is because many packets collide when the retransmission interval is short, lowering throughput, but as the interval becomes longer, packet collisions become less common. However, if the interval is too long, the retransmission timing becomes late, the connection response becomes poor, and the TCP performance deteriorates. Therefore, we must set the interval carefully. In this topology, an interval of 3-17 time slots allowed the best throughput without the delayed ACK option and our proposed technique improved throughput by 60%. With the delayed ACK option, the best throughput was obtained when the interval was 3-13 time slots; in this case, throughput was



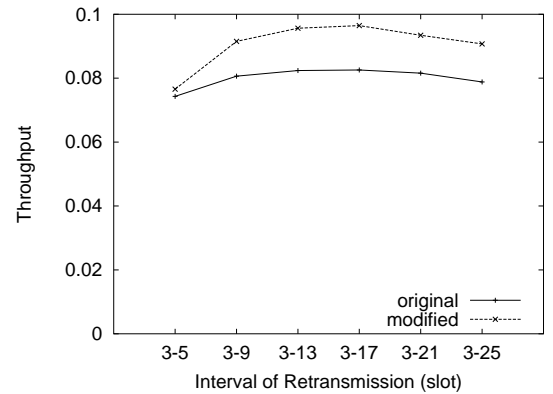
(a) Without the delayed ACK option



(a) Without the delayed ACK option



(b) With the delayed ACK option



(b) With the delayed ACK option

**Fig. 5** Throughput of one connection on chain topology

**Fig. 6** Throughput of crossing two connections on cross-chain topology

improved by 20%. The throughput with the delayed ACK option was better than without the option entirely, because the option reduced packet collision in the network. The rate of improvement with the option was lower, though, because packet collision had already been suppressed by the option.

Second, we used the cross-chain topology (Fig. 4(b)). In this topology, we observed the effect of collisions when there were two different connections. (The connection between nodes 5 and 8 was added to the connection shown in Fig. 4(a).) The throughput shows the added value of throughput of two connections. The results for this topology are shown in Fig. 6. The general pattern of the results was similar to that for the chain topology, but the rate of improvement with our proposed technique was only 20% without the delayed ACK option and 15% with the option - slightly lower than for the chain topology. Our proposed technique focuses on packet collisions caused by one TCP connection, and if there are more than one connection in the network, cross connections occur. Compared with a

single-connection case, the degree of improvement thus becomes smaller.

Next, we simulated three or more connections in the mesh network (Fig. 4(c)). The three connections were nodes 0 to 4, nodes 13 to 18, and nodes 15 to 16. The results are shown in Fig. 7. The pattern again resembled the previous cases with an 18% improvement without the delayed ACK option and a 10% improvement with the option. Last, we simulated random connections with the mesh topology. We set the number of TCP connections (i.e., the network load) to 3, 6, or 9. Random connection meant that two nodes were randomly selected and one became a sender while the other was a receiver. We generated 20 connection patterns and averaged the rate of improvement. Since we had obtained the best throughput using 3-17 time slots in the earlier simulations, we used 3-17 time slots here. The results are shown in Table 1. When there were 9 connections, the network load was very high, but the rate of improvement was at least close to 10%. Thus,

**Table 1** Average percentage of improvement at the time of setting up a connection at random on mesh topology

load	delayed ACK option	
	OFF	ON
3	23.47	21.53
6	15.86	12.96
9	12.38	9.93

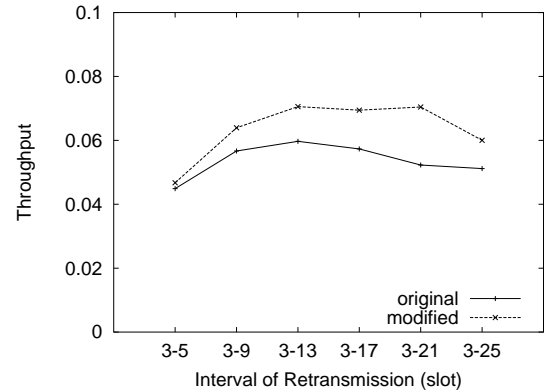
even when the load is very high and the connections are random, our proposed technique is effective in FRN.

Finally, we investigate the influence which the distance (the number of hop) between a source node and a destination node has on the proposed technique. It is considered that the effect of the proposed technique is influenced with a connection's length. That is, since the response time of TCP becomes short when distance is short, the case where a packet exists in a middle node increases. Therefore, the case where combination of data and ACK packet is performed increases and it is expected that the effect of the proposed technique becomes large. We changed distance variously from 3 hop to 12 hop in chain topology, and performed the simulation. The time slot of retransmission was similarly set to 3-17 here. We show change of the throughput by changing a connection's distance at the time making delayed ACK option On and Off in Fig. 8, and show each rate of an improvement in Fig. 9. These results indicate that the one where a connection's distance is shorter has the large effect of the proposal technique. However, it is also revealed that the effect of proposed technique is large enough even when a connection's distance is long.

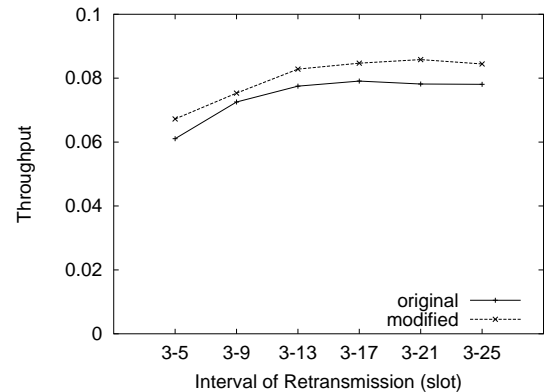
#### 4.3 Applicability of Our Approach to Generic Ad Hoc Networks

In this paper, we have proposed a technique that is able to improve performance of TCP over an ad hoc network by combining a data packet and an ACK packet. We have solved the problem of frequent packet collisions, which occurs due to the reverse directions of TCP data and ACK packet streams, by combining those packets. We evaluated the proposed technique for a specific ad hoc network, FRN [14, 17]. FRN adopts a table-driven type routing protocol and is characterized by the receipt check of packets by the echo as mentioned above. However, the proposed technique itself does not depend on FRN, and it is easily applicable not only to FRN but also to generic ad hoc networks as explained below.

For our proposed technique to be applicable, the following two mechanisms should be equipped with the underlying ad hoc networks. First, it is necessary to use broadcasting at the time of packet transmission for the two different destinations of data and ACK packets to simultaneously receive those packets. Second, each node has to distinguish between data packets and ACK packets, and manage them separately. For that



(a) Without the delayed ACK option

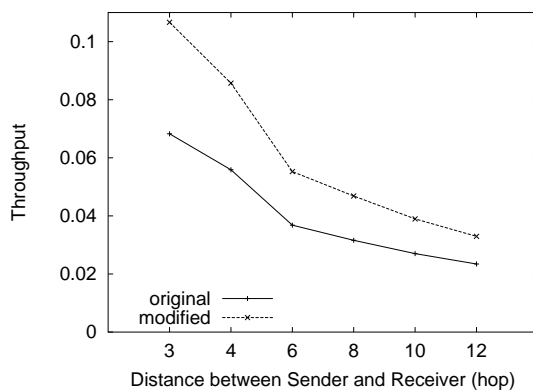


(b) With the delayed ACK option

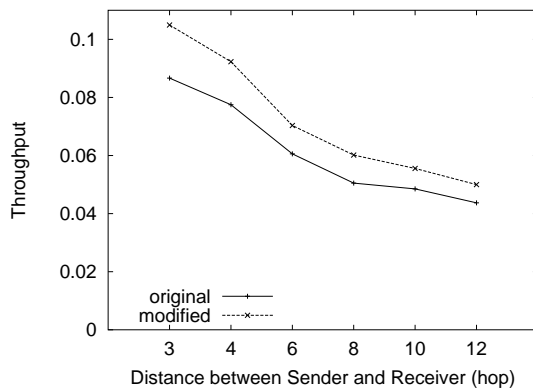
**Fig. 7** Throughput of three connections intermingled on mesh topology

purpose, the data link layer protocol needs to distinguish whether a packet is data or ACK of TCP layer. Then, data and ACK packets are combined. However, we should note here that while we described in this paper that we manage data packets and ACK packets at different queues, our technique is realizable even if both packets are queued in the same buffer. Only a requirement is that each can be distinguished and can be retrieved from the buffer.

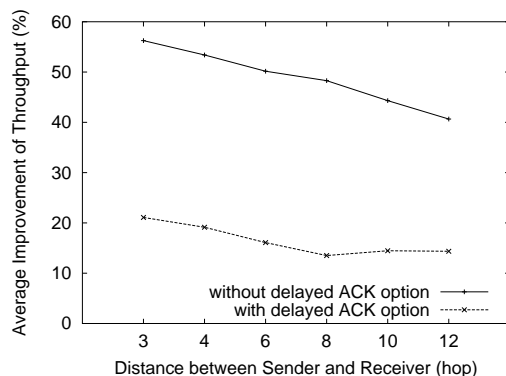
Our proposal does not require other restrictions. The routing protocol in an ad hoc network including DSR (Dynamic Source Routing), AODV, and DSDV, can be used with our proposal. Furthermore, our proposal does not depend on the versions of TCP. Therefore, our proposal can be easily applied to generic ad hoc networks, in order to improve the performance of TCP, while numerical validations are left to be a future research topic.



(a) Without the delayed ACK option



(b) With the delayed ACK option

**Fig. 8** Throughput of one connection on chain topology while changing the distance**Fig. 9** Average improvement of one connection on chain topology while changing the distance

## 5. Conclusion and Future Work

In this paper, we analyzed the problem of packet collision in an ad hoc network resulting from the bidi-

rectional communication of TCP. We have proposed a technique that combines data and ACK packets, and have shown through simulation that this technique can make radio channel utilization more efficient. In the simulation, the technique improved the TCP performance by up to 60%, and by about 10% even when the network load was very high. It can be expected that the method which we proposed here is effective not only in a specific ad hoc network but the general ad hoc network which the collision of bidirectional communication generates.

In the future, we will analyze a routing method that prevents packets from colliding. This method will enable data packets and ACK packets to pass along separate paths, thus eliminating the possibility of collision between data packets and ACK packets. And we will study the method of raising the performance of TCP from the lower layer of a transport layer.

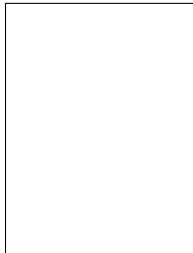
## Acknowledgements

This work was partly supported by the Special Coordination Funds for promoting Science and Technology of the Ministry of Education, Culture, Sports, Science and Technology of Japan.

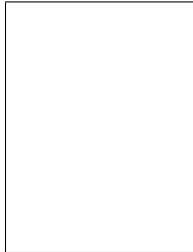
## References

- [1] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley, 1994.
- [2] D. Kim, C. K. Toh, and Y. Choi, "TCP-BuS: Improving TCP performance in wireless ad hoc networks," in *Proc. ICC 2000*, June 2000.
- [3] G. Holland and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks part I: Problem discussion and analysis of results," in *Proceedings of the ACM/IEEE MOBICOM'99*, Aug. 1999.
- [4] S. Bansal, R. Shorey, S. Chugh, A. Goel, K. Kumar, and A. Misra, "The capacity of multi-hop wireless networks with TCP regulated traffic," in *Proceedings of the GLOBECOM 2002*, Nov. 2002.
- [5] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP throughput and loss," in *Proc. INFOCOM 2003*, Mar. 2003.
- [6] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A feedback-based scheme for improving TCP performance in ad hoc wireless networks," *IEEE Personal Communications Magazine*, vol. 8, pp. 34–39, Feb. 2001.
- [7] J. Liu and S. Singh, "ATCP: TCP for mobile ad hoc networks," *IEEE Journal of Selected Areas in Communications*, vol. 19, pp. 1300–1315, July 2001.
- [8] Z. Fu, B. Greenstein, X. Meng, and S. Lu, "Design and implementation of a TCP-friendly transport protocol for ad hoc wireless networks," in *Proc. IEEE ICNP'02*, Nov. 2002.
- [9] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar, "ATP: A reliable transport protocol for ad hoc networks," in *Proc. MOBIHOC2003*, June 2003.
- [10] T. Yuki, T. Yamamoto, M. Sugano, M. Murata, H. Miyahara, and T. Hatauchi, "A study on performance improvement of TCP over an ad hoc network," *IEICE Trans. Commun. (Japanese Edition)*, vol. J85-B, pp. 2045–2053, Dec.

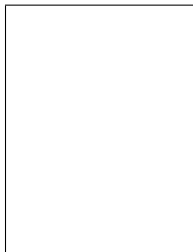
- 2002.
- [11] S. Ray, J. Carruthers, and D. Staorobinski, "RTS/CTS-Induced congestion in ad hoc wireless LANs," in *Proc. IEEE WCNC'03*, Mar. 2003.
  - [12] "Flexible Radio Network, Fuji Electric Co. Ltd." available at [http://www.fujielectric.co.jp/denki/p26/ecop\\_contents2.html](http://www.fujielectric.co.jp/denki/p26/ecop_contents2.html).
  - [13] C. E. Perkins, *Ad Hoc Networking*. Addison-Wesley, 2001.
  - [14] M. Sugano, T. Araki, M. Murata, T. Hatauchi, and Y. Hosooka, "Performance evaluation of a wireless ad hoc network: Flexible radio network (FRN)," in *Proc. IEEE ICPWC 2000*, pp. 350-354, Dec. 2000.
  - [15] "The Network Simulator - ns-2." available at <http://www.isi.edu/nsnam/ns/>.
  - [16] "The CMU monarch project." available at <http://www.monarch.cs.cmu.edu/>.
  - [17] T. Yamamoto, M. Sugano, M. Murata, T. Hatauchi, and Y. Hosooka, "Performance improvement of an ad hoc network system for wireless data service," *IEICE Trans. Commun.*, vol. E86-B, pp. 3559-3568, Dec. 2003.



**Taich 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.



**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.



**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. Since September 1996, he has been Associate Professor in Osaka Prefecture College of Health Sciences. 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 Advanced Networked Environment Division, Cybermedia Center, Osaka University in April 2000. He has more than two 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.



**Hideo MIYAHARA** received the M.E. and D.E. degrees from Osaka University, Osaka, Japan in 1969 and 1973, respectively. From 1973 to 1980, he was an Assistant Professor in the Department of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University, Kyoto, Japan. From 1980 to 1986, he was an Associate Professor in the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University, Osaka, Japan. From 1986 to 1989, he was a Professor of the Computation Center, Osaka University. Since 1989, he has been a Professor in the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University. From 1995 to 1998, he was a director of Computation Center of Osaka University. From 1998 to 2000, he was a dean of the Faculty of Engineering Science, Osaka University. From 2002 to 2003, he is a Dean of Graduate School of Information Science and Technology, Osaka University. From 2003, he is the president of Osaka University. From 1983 to 1984, he was a Visiting Scientist at IBM Thomas J. Watson Research Center. His research interests include performance evaluation of computer communication networks, broadband ISDN, and multimedia systems. Prof. Miyahara is a Fellow of IPSJ and an IEEE Fellow.



**Takaaki HATAUCH** 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.