Master's Thesis

Title

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

Supervisor

Professor Hideo Miyahara

Author

Taichi Yuki

February 13th, 2004

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

A Study on Data Link Layer Control Methods

for Performance Improvement of TCP over an Ad Hoc Network

Taichi Yuki

Abstract

In recent years, the applying area of ad hoc networks has expanded, and demands for interconnection with wired networks and the same service offered in wired networks have risen. In the integration of ad hoc networks and wired networks, TCP (Transmission Control Protocol) will be used as a transport layer protocol because TCP is generally used in wired networks. However, when TCP is used in an ad hoc network, it is known that the throughput of TCP will deteriorate because of the high error rate of wireless channels and the collisions of packets. Many other studies convert TCP itself, or propose a new transport layer protocol to solve these problems. However, we do not want to modify TCP itself because seamless communication between wired networks and ad hoc networks is desirable. Therefore, in order to increase TCP performance in an ad hoc network, support from the lower layers of the transport layer is required.

In this thesis, we propose two techniques that approach from the data link layer. First, we focus on the frequent collisions of the opposite direction's packets of data and ACK. We reduce the collisions of packets by combining a data packet and an ACK packet and transmitting them simultaneously. With this technique, we can increase the efficiency of wireless channel utilization, reduce the number of packet collisions, and improve the performance of TCP. We have found that throughput can be improved by up to 60% by applying this technique. Moreover, 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. Therefore, we propose a second technique of performing retransmission of the data link layer efficiently according to the load of the nodes. Furthermore, we evaluate its performance by simulations and show that the performance of TCP improves by the proposed techniques. We have found that throughput can be improved by up to 16% with this technique and that this technique is effective when wireless channel error occurs.

Keywords

Ad hoc networks

Collisions

TCP (Transmission Control Protocol)

Data link layer

Retransmission

Throughput

Simulation

Contents

List of Figures

List of Tables

1 Introduction

With the progress of wireless communications technology, a wireless network that communicates through radio waves has been put into practical use and its application to various fields has spread. Among systems that build wireless networks, a cellular communication system connects a terminal through a base station connected to a wired network like a cellular phone or wireless LAN. On the other hand, an ad hoc network is a system that can build a self-organized network because wireless terminals communicate with each other and exchange network information over the wireless channel. These terminals are able to relay packets for another terminal and can form a wide-area multi-hop wireless network. Since an ad hoc radio network needs neither a base station nor a wired circuit, it does not need to wire at the time of arrangement of a terminal or network extension, and can respond to dynamic change of the network as a result of failure, movement, etc., of a terminal by autonomous operation of each terminal. Therefore, analysis of the characteristics of ad hoc networks and research on routing protocols are now done briskly [1-9].

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) [10] 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.

Many studies to the improvement of TCP performance have been dedicated over ad hoc networks [6, 11-17]. Much of this research has focused on TCP performance degradation caused

by terminal movement. For example, [6] 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 [6] 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 [18]. [14] and [15] produce techniques that distinguish route failure and congestion, and decide whether to control congestion or not. Therefore, they can cope with situations correctly. In [16] and [17], new transport layer protocols that are suitable for ad hoc networks are introduced. However, in this thesis, we do not modify TCP because we regard the seamless communication between wired networks and ad hoc networks as the most important subject.

On the other hand, 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 channels. As packets travel over multi-hop wireless links, they often collide. The IEEE 802.11 standard provides for channel reservation based on an RTS/CTS (request to send/clear to send) 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 [19].

First, 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 simul-

taneously. In this way, packet collisions are avoided and effective use of a wireless channel is achieved.

To evaluate this technique, we applied it to an ad hoc network that uses table-driven routing with fixed terminals. 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. 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 [20, 21]. 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.

Moreover, retransmission by the data link layer 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 second 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. We again use the FRN system to evaluate this proposal.

This thesis is organized as follows. In Section 2, we begin by describing the Flexible Radio Network (FRN). We introduce the outline of system and protocol of FRN. In Section 3, we investigate the problem of collisions of data packets and ACK packets of TCP, propose a technique to reduce the effect of the problem, and then evaluate this technique by means of simulation. In

Section 4, furthermore, we investigate the problem of collisions of packets and the high incidence of duplicate packets due to the high load of the network. Therefore, we focus on a packet retransmission control mechanism and propose a technique to reduce the effect of the problem. Then, we evaluate this technique by means of simulation. Finally, we conclude this thesis and outline several remaining research topics in Section 5.

2 Target Ad Hoc Network System

2.1 System Description

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 developed by the Fuji Electric Company. It is a data-collection system that has been used as a kind of sensor network. Early FRN applications have included power consumption data collection, usage information collection from ski-lift gates, and sales account collection and monitoring of vending machines.

The composition element of FRN is shown in Figure 1. In the FRN, every wireless terminal is called a *node*. Nodes with which a node can communicate directly are called *neighbor nodes*. Every node is able to select a route for a packet and relay the packet to a neighbor node. Specifically, a *host node* that links to the Data Terminal Equipment (DTE: Data Terminal Equipment) and generates and receives data packets and other nodes called *relay nodes* make up a multi-hop network. Every node maintains the network information in a *configuration table* that contains the routing information from the node to each destination node (Table 1). The routing information consists of a list of the neighbor nodes on the route to the destination node and the hop count of the route (Table 2). Each node obtains the number of times of the shortest relay from a network

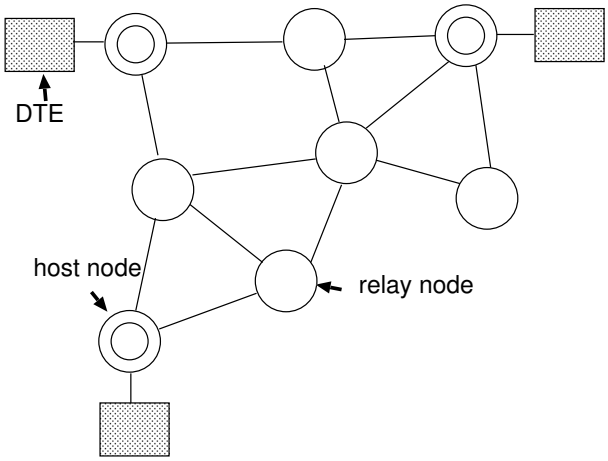


Figure 1: Structure of FRN

Table 1: Network configuration table

	Dest. Node 0	Dest. Node 1	...
Route 1	Neighbor Info.	Neighbor Info.	...
Route 2	Neighbor Info.	Neighbor Info.	...
⋮	⋮	⋮	⋱

Table 2: Neighbor information

Neighbor node's ID in order to reach the destination node
The number of times of relay from the neighbor node to the destination node
The reliability of the wireless channel to the neighbor node

configuration table to all nodes, generates a configuration control packet as shown in Table 3, and transmits to the neighbor nodes, without specifying a destination node, for every fixed cycle called configuration control cycle. The node which received the configuration control packet reconstructs an own network configuration table based on the information.

The reliability of the wireless channel in the neighbor information in Table 2 is a value showing the success probability of communication between nodes, and it is managed as a reliability managed table as shown in Table 4. Each node holds the receiving situation of the configuration control packet of past X time as a reliability value like Y/X, and □ and $\times$ mean receiving and un-receiving of the configuration control packet from other nodes. Moreover, whether nodes recognize it as direct communication being possible sets a threshold to a reliability. If the reliability

Table 3: An example of a configuration control packet

ID:0	ID:1	ID:2	ID:3	...
$0^{\pm}$	2	1	3	...

$^{\pm}$Its own number of times of relay is set to 0.

Table 4: An example of reliability managed table

ID:0	□ × □ □ ‐ □ ×
ID:1	× × □ □ ‐ × ×
⋮	⋮
⋮	⋮

to the node is larger than the threshold, the node will be recognized as a neighbor node and it will be registered in a network configuration table. On the other hand, if the reliability to the node is smaller than the threshold, the node will not be recognized as a neighbor node.

The FRN's routing system is table-driven, like a DSDV (destination-sequenced distance vector) type [1], with periodic communication. Routing protocols of an on-demand type, such as AODV (ad-hoc on-demand distance vector) [1], 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.

2.2 Protocol Description

2.2.1 Data–link Protocol

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

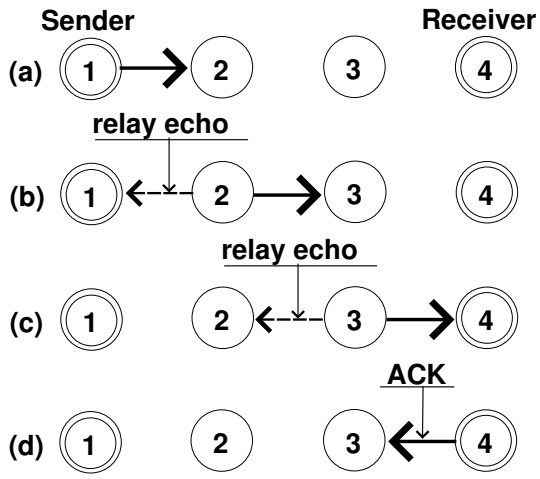


Figure 2: Relay echo mechanism

a FRN ACK are done in the time slot immediately after the slot in which the node receives the packet. Figure 2 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 thesis, we tentatively set it as 32 slots.

2.2.2 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 is as follows. For each destination, routes are classified into three groups according to their hop count (Figure 3).

- 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.

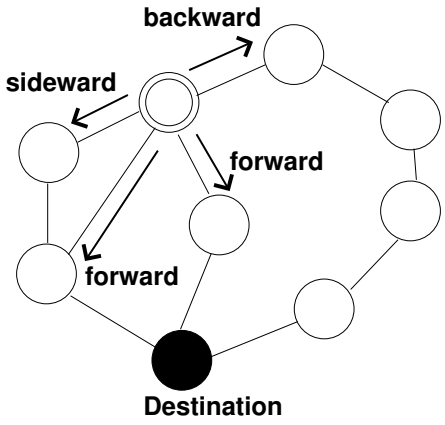


Figure 3: Classification of neighbor nodes

3 Technique to Prevent Collisions of TCP Data and ACK Packets

3.1 Problems in Bidirectional Communication of TCP

From our past research [18], we found that simply avoiding link failure was not sufficient to improve TCP performance, and the collisions of packets become big problem. 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 [19].

In this thesis, 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.

3.2 Proposed Technique

We explained the problems that arise in ad hoc networks in Section 3.1. In this thesis, 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 wireless 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 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.

Figure 4 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 (Figure 4(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

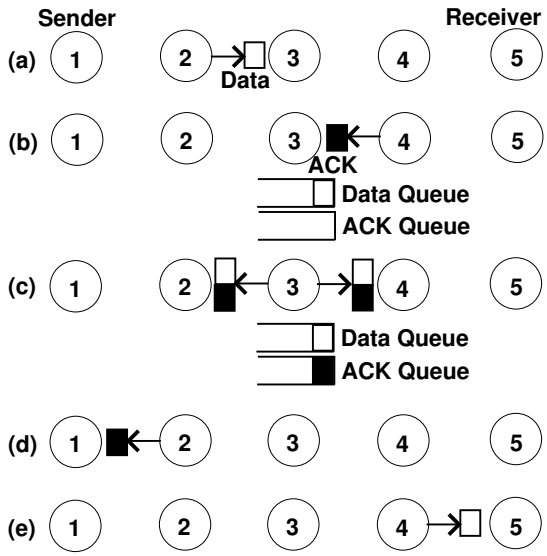


Figure 4: A process where the proposed technique is used

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. Figure 4(d) and 4(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 wireless 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 [10]. 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 the FRN can be realized. Therefore, we next evaluated whether our proposed technique is also effective when used simultaneously with this option.

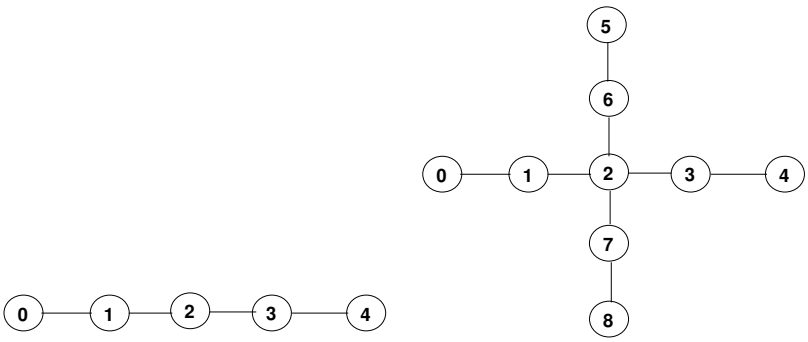
3.3 Simulation and Evaluation

3.3.1 Simulation Model

We evaluated our proposed technique through simulations using ns–2 [22] with its radio propagation model extended by the CMU Monarch Project [23]. 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.

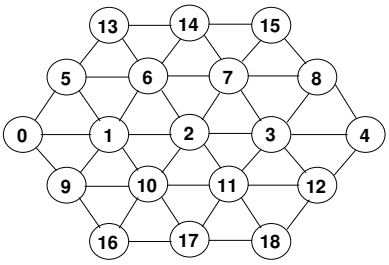
We used the network topologies shown in Figure 5. A circle and a number in the circle mean a node and its address. A line connecting two nodes means that they can communicate directly. Although these topologies are very simple, we can use them to identify basic tendencies of our proposed technique and 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 Figure 5(a) topology to evaluate the technique in a pure bidirectional connection. In Figure 5(b), there is a crossing of connections and we evaluated the technique in such a case. Figure 5(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.



(a) Chain topology

(b) Cross-chain topology



(c) Mesh topology

Figure 5: Network topologies

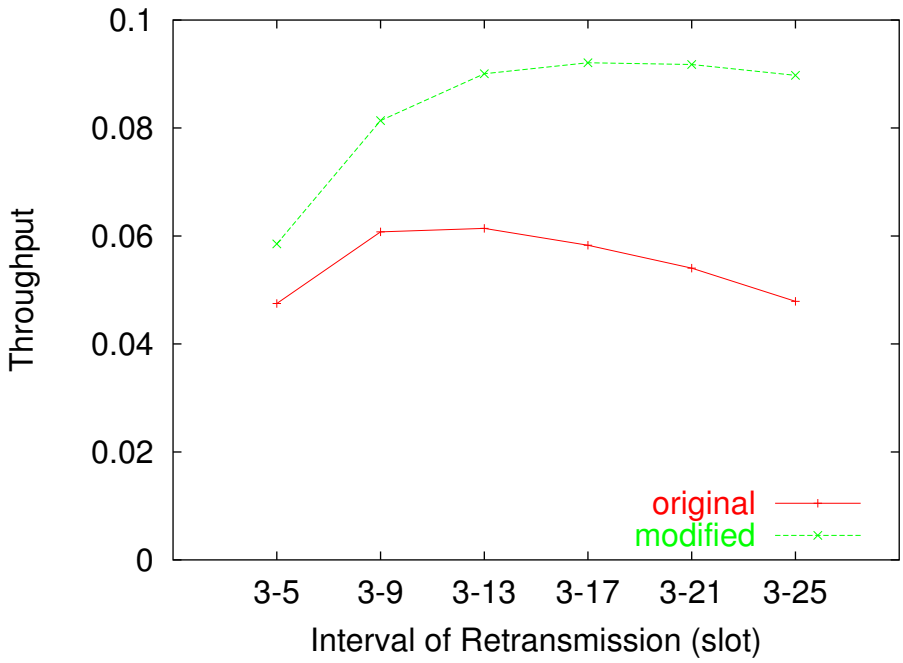
3.3.2 Evaluation Results

First, we evaluated throughput with the simplest topology (Figure5(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 Figure6. As mentioned (Section 2), we changed the retransmission interval from 3-5 time slots to 3-25 time slots and measured the throughput at each interval. Figure 6(a) and Figure 6(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

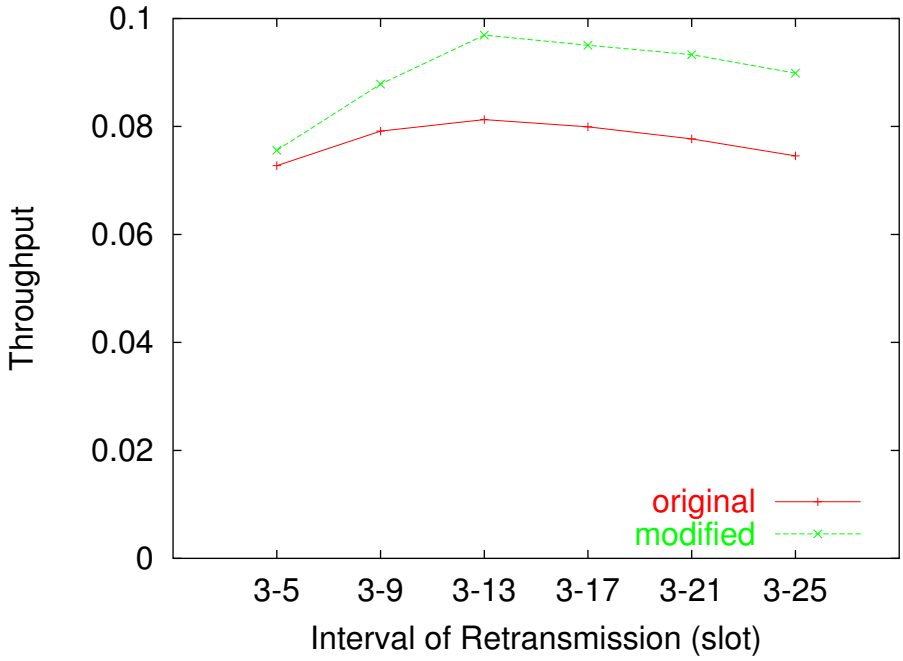
is too long, the retransmission timing becomes late, the connection response becomes poor, and the TCP performance deteriorates. Therefore, we must set the interval carefully. In this topology, an interval of 3-17 time slots allowed the best throughput without the delayed ACK option and our proposed 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 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 (Figure 5(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 Figure 5(a).) The throughput shows the added value of throughput of two connections. The results for this topology are shown in Figure 7. 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 (Figure 5(c)). The three connections were nodes 0 to 4, nodes 13 to 18, and nodes 15 to 16. The results are shown in Figure 8. The pattern again resembled the previous cases with an 18% improvement without the delayed ACK option and a 10% improvement with the option. Moreover, 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 5. When there

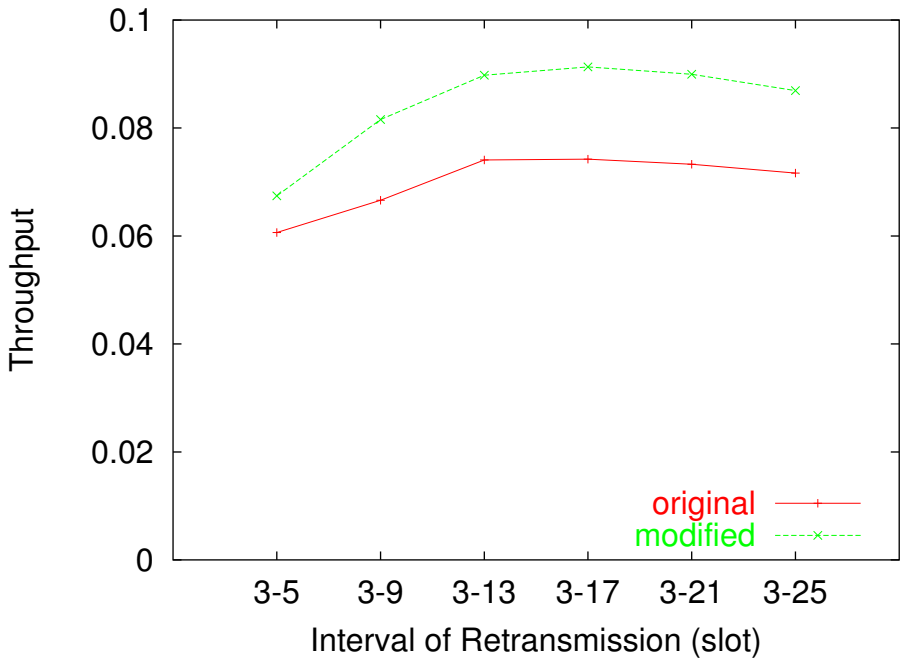


(a) Without the delayed ACK option

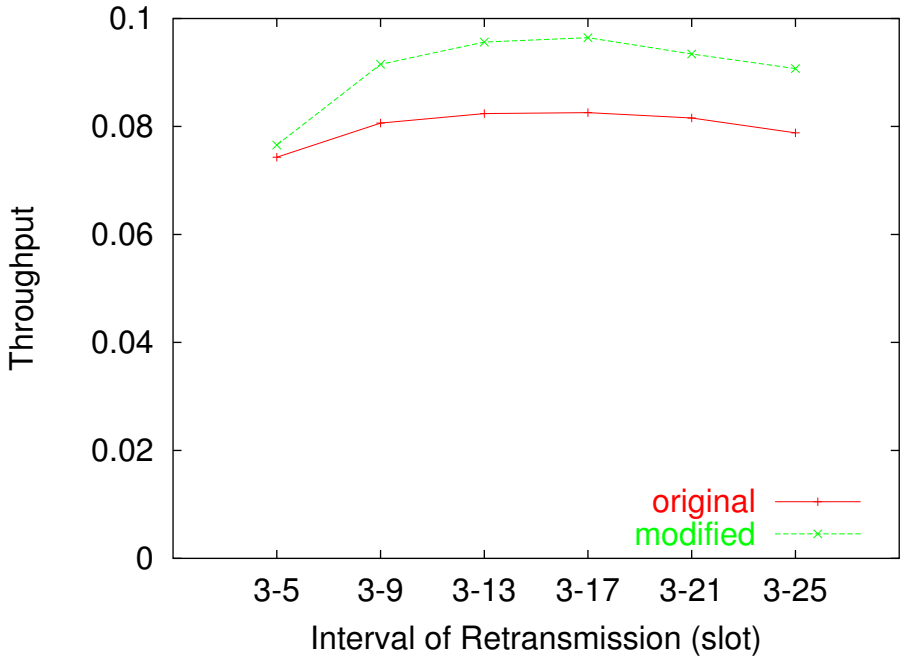


(b) With the delayed ACK option

Figure 6: Throughput of one connection on chain topology



(a) Without the delayed ACK option



(b) With the delayed ACK option

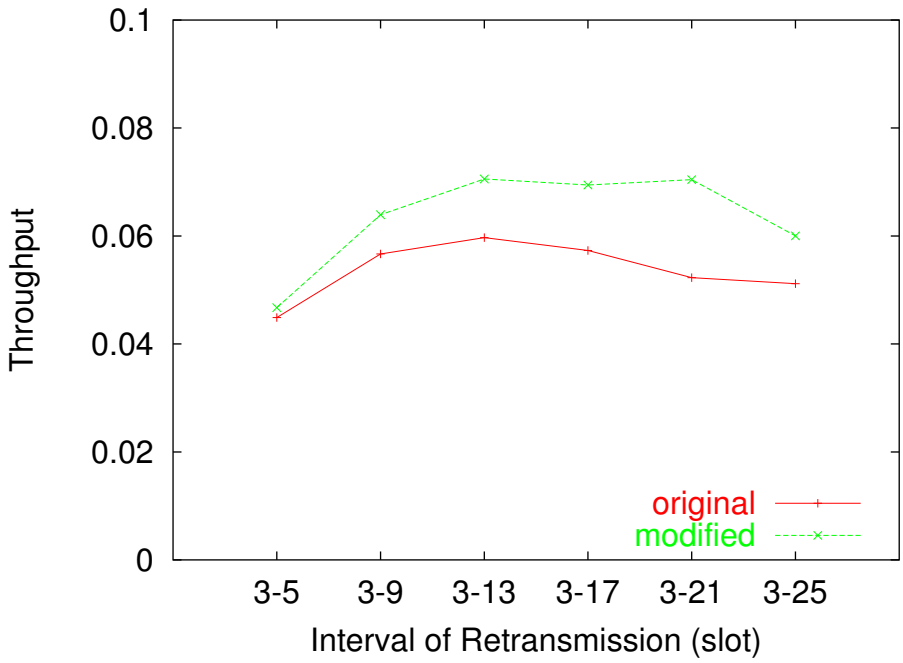
Figure 7: Throughput of crossing two connections on cross-chain topology

Table 5: Average percentage of improvement at the time of setting up a connection at random on mesh topology

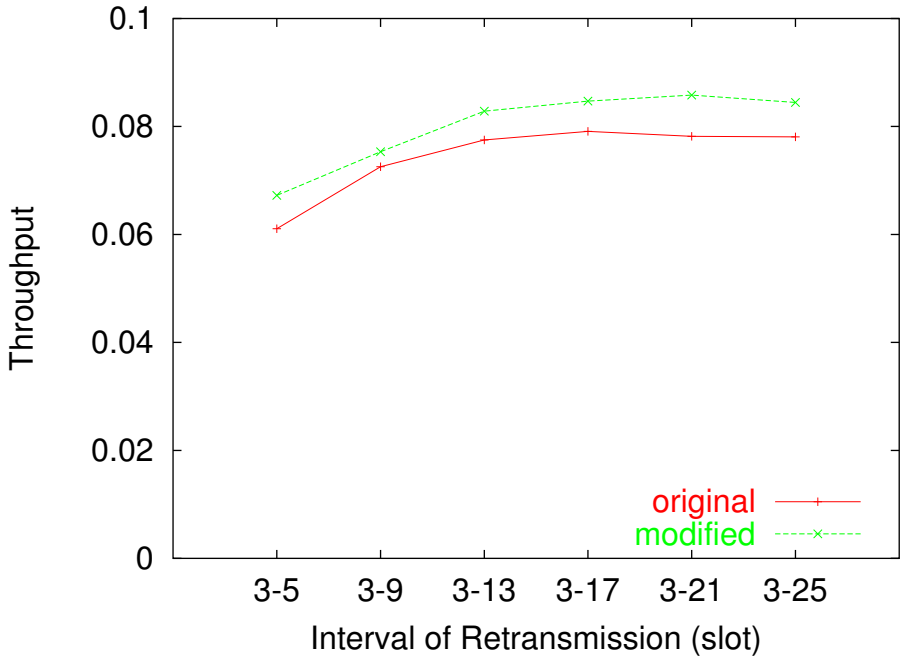
load	delayed ACK option	
	OFF	ON
3	23.47	21.53
6	15.86	12.96
9	12.38	9.93

were 9 connections, the network load was very high, but the rate of improvement was at least close to 10%. Thus, 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 Figure 9, and show each rate of an improvement in Figure 10. 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.

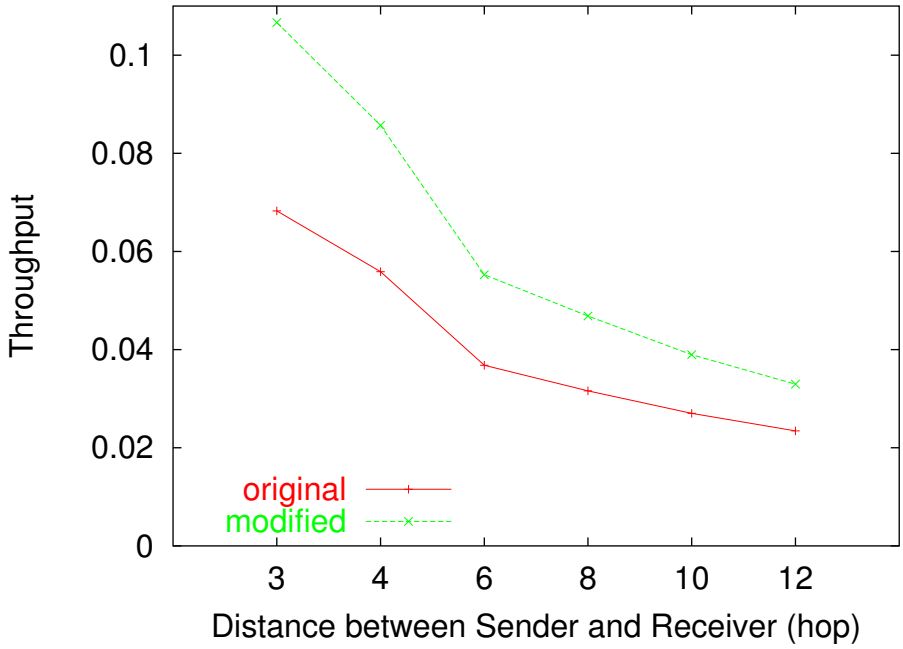


(a) Without the delayed ACK option

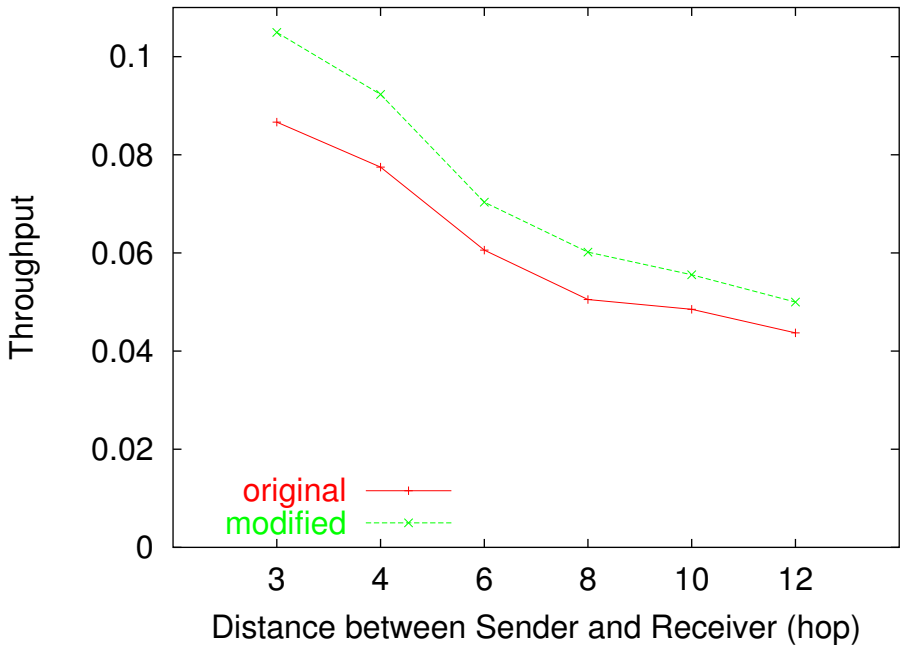


(b) With the delayed ACK option

Figure 8: Throughput of three connections intermingled on mesh topology



(a) Without the delayed ACK option



(b) With the delayed ACK option

Figure 9: Throughput of one connection on chain topology while changing the distance

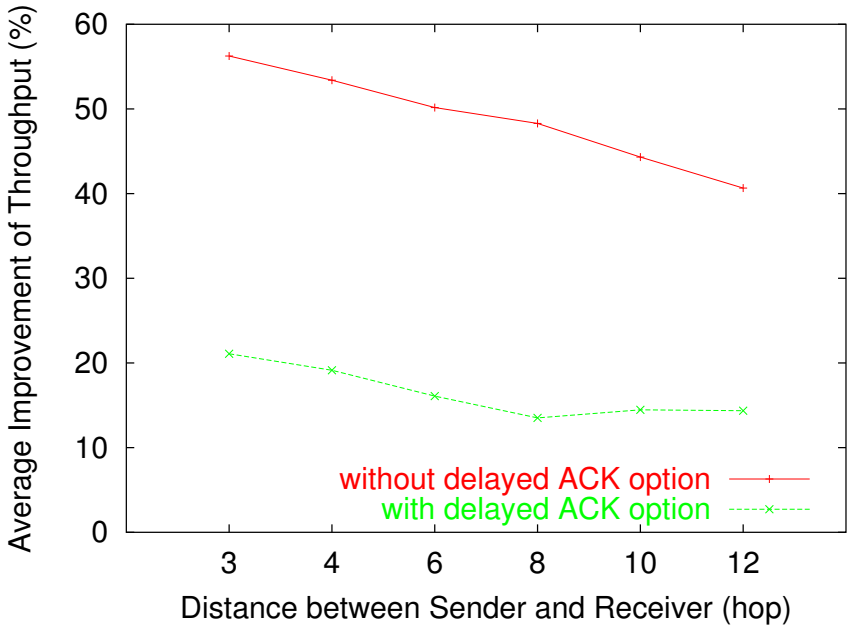


Figure 10: Average improvement of one connection on chain topology while changing the distance

4 Technique to Control Packet Retransmission

4.1 Problems of Packet Collisions in High Load Situations

In the preceding section, we focused on collisions of data packets and ACK packets of TCP, and proposed a technique for it. However, when TCP is used over an ad hoc network, the problem of other packet collisions still arises. 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 11. Figures 11(a) and 11(b) show that a packet is forwarded to a destination node. However, if a relay echo of the packet collides with another packet (Figure 11(c)), the node will not know that the packet could be forwarded correctly and will forward the packet to another node again (Figure 11(e)). In addition, in Figure 11(f), duplicate packets exist in the network. Therefore, once the load increases, it will continue to increase and the network will fail eventually. Although

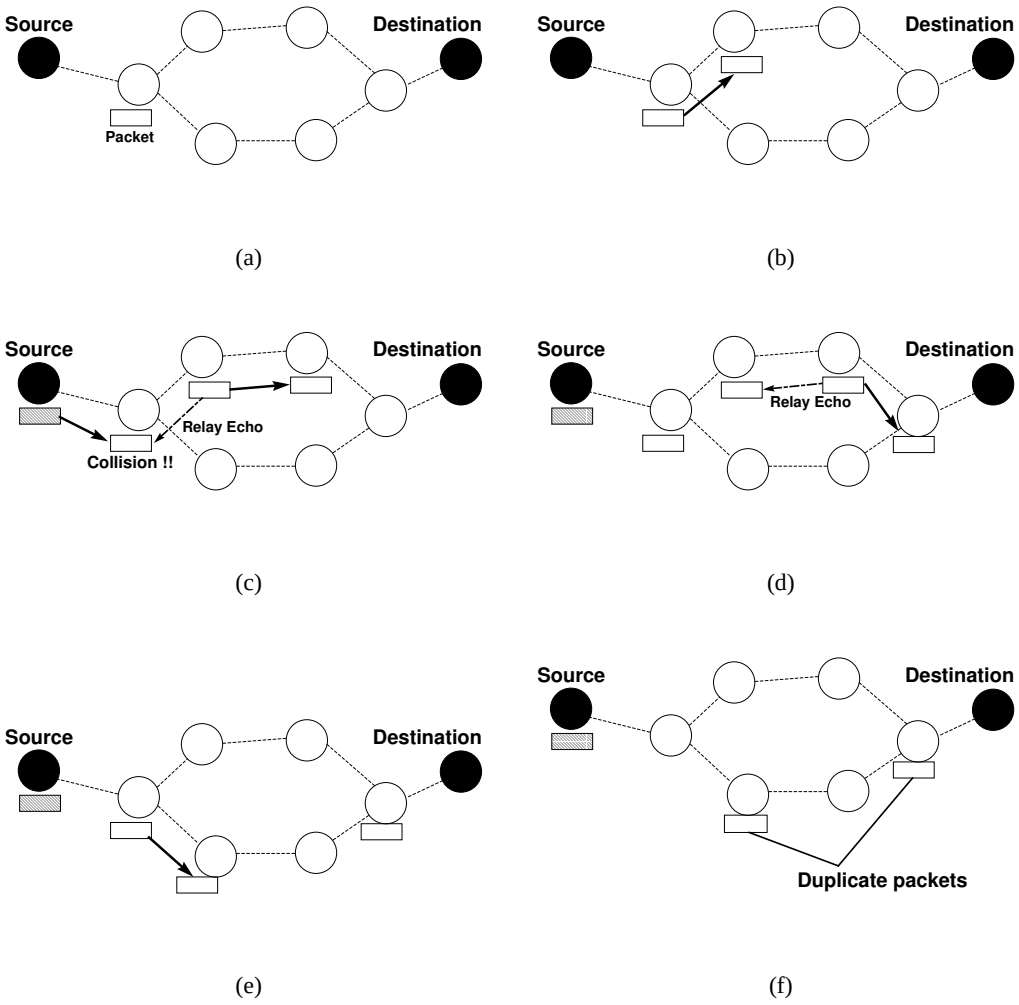


Figure 11: An example of a process of duplicate packets generation

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.

4.2 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 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 6. 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 12. 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.

Table 6: 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	□	×	□	‥

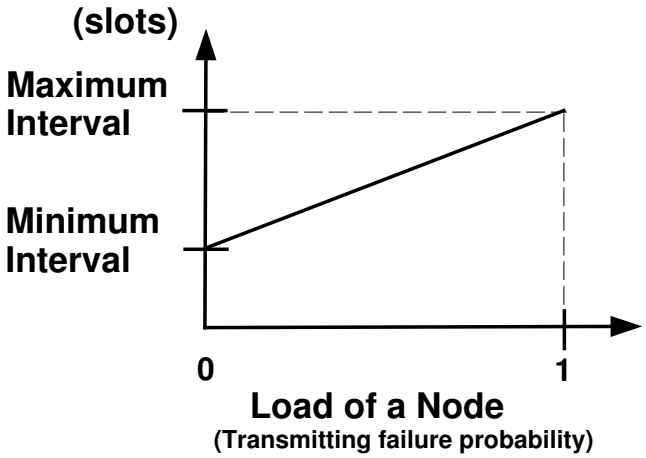


Figure 12: Changing the retransmission interval by load

Next, we describe the matter that must be changed in order to use this proposal. Until now, a packet was discarded when the set-up life time reached zero, as we stated in Section 2.2.1. However, 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 11) 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.

4.3 Simulation and Evaluation

4.3.1 Simulation Model

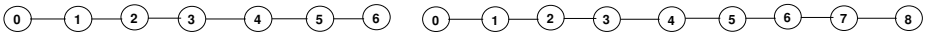
As described in Section 3.3.1, we also use ns–2 as the simulator. The other fundamental parameters are almost the same. We also use throughput as a measure of performance.

In addition to Figure 5, we used the network topology of Figure 13 in order to observe the influence of the proposal technique in the connection of a large hop. 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.

4.3.2 Evaluation Results

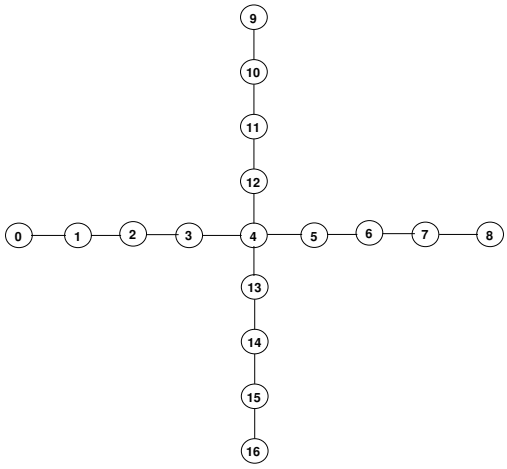
4.3.2.1 Evaluation of Maximum Count of Retransmission

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 (Figure 5(a)), 6 hop (Figure 13(a)), and 8 hop (Figure 13(b)). By using these topologies, we un-



(a) 6-hop chain topology

(b) 8-hop chain topology



(c) 8-hop cross-chain topology

Figure 13: Newly added network topologies

derstood 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%.

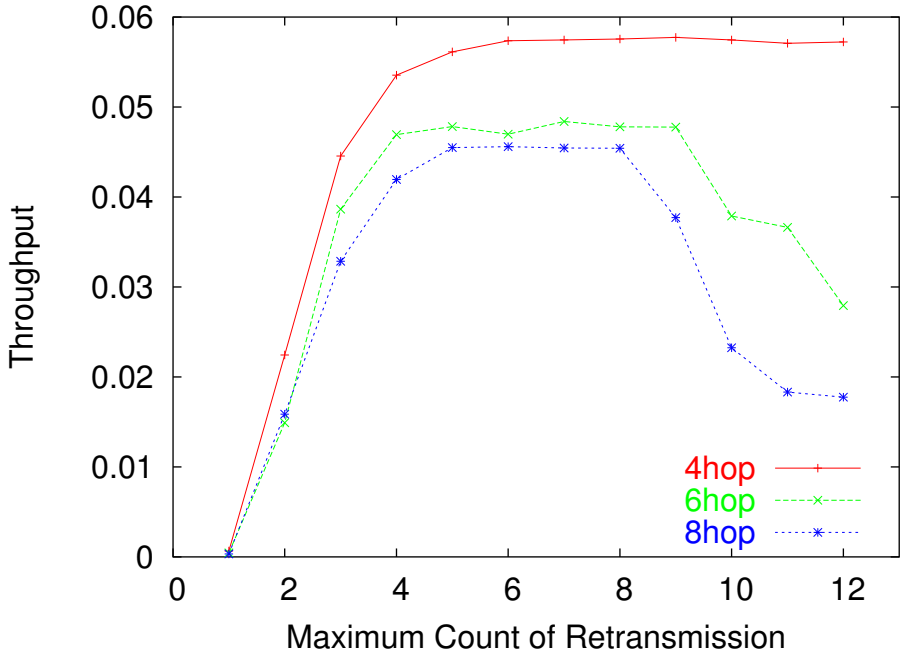
We show change of the throughput by the count of maximum retransmission when not generating the random error of a wireless channel in Figure 14. Figure 14(a) shows change of the throughput when the retransmission interval is set as 3-5 time slots. In this figure, if the maximum count of retransmission 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 14(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 15 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 15(a)). Moreover, from Figure 15(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.

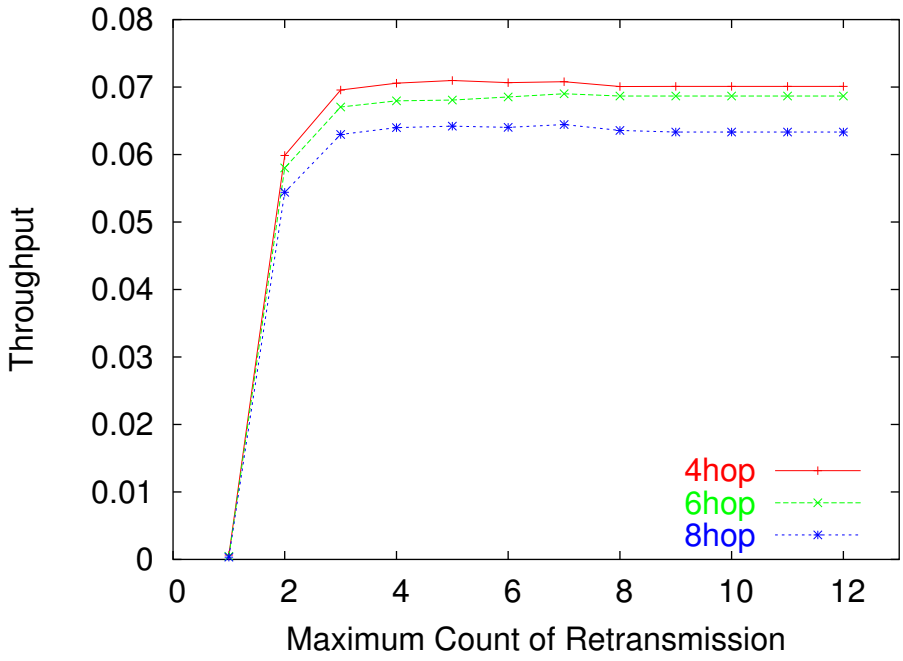
Although we considered it better for the connection with a large hop number to make a large maximum count of retransmission, it is shown that this is not correct. The connection with a large hop number tends to generate collisions of packets, and also tends to generate packet pile ups. This result showed 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 retransmission at 4 and show the throughput when changing the dispersion of a retransmission interval in Figure 16. Figure 16(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 16(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 other various topologies (Figure 5(b), Figure 5(c), and Figure 13(c)). The results are shown in Figure 17

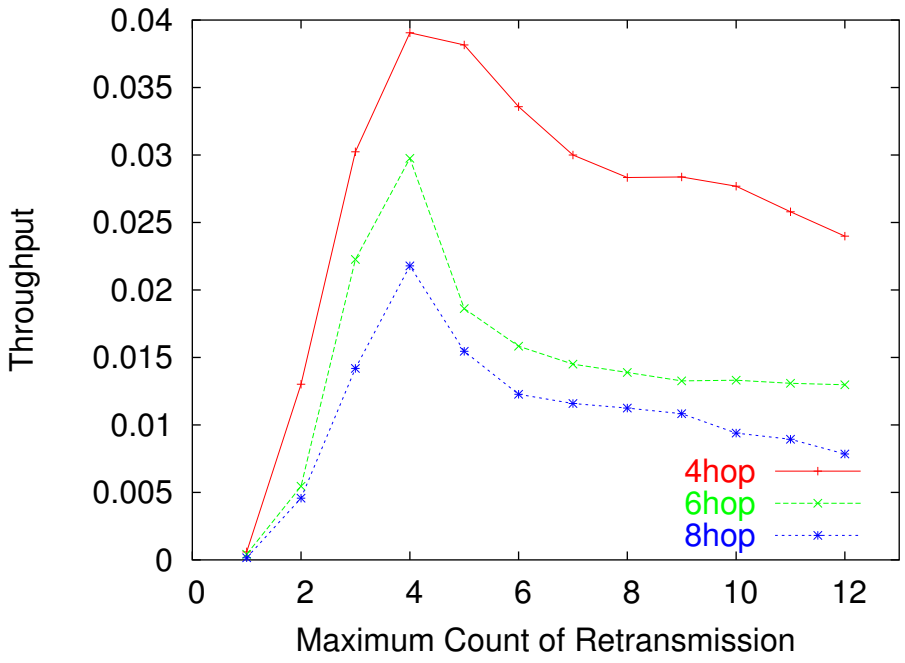


(a) Interval of retransmission: 3-5 time slots

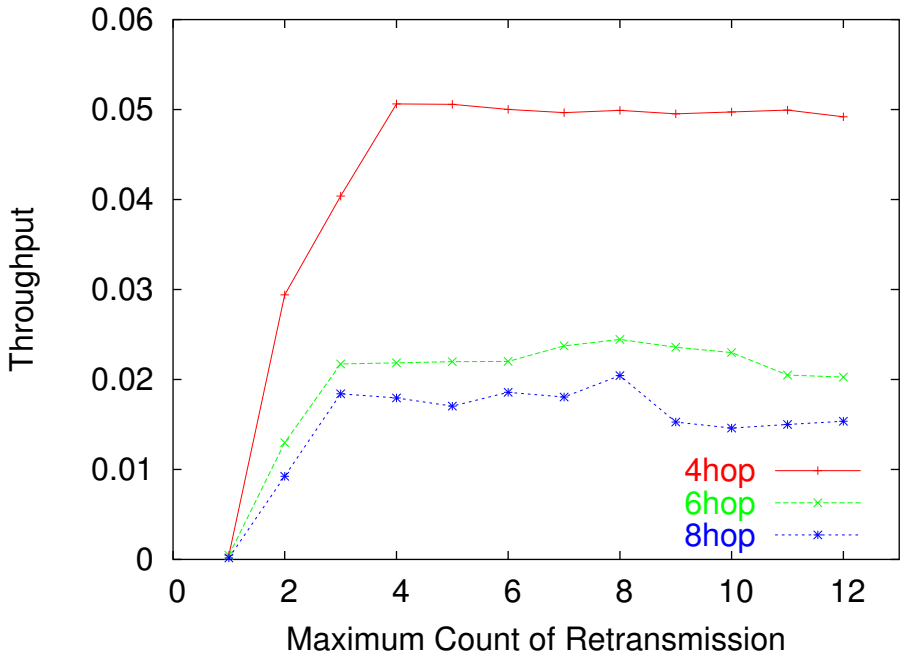


(b) Interval of retransmission: 3-17 time slots

Figure 14: Change of the throughput according to a maximum count of retransmission without the error of a wireless channel in chain topologies

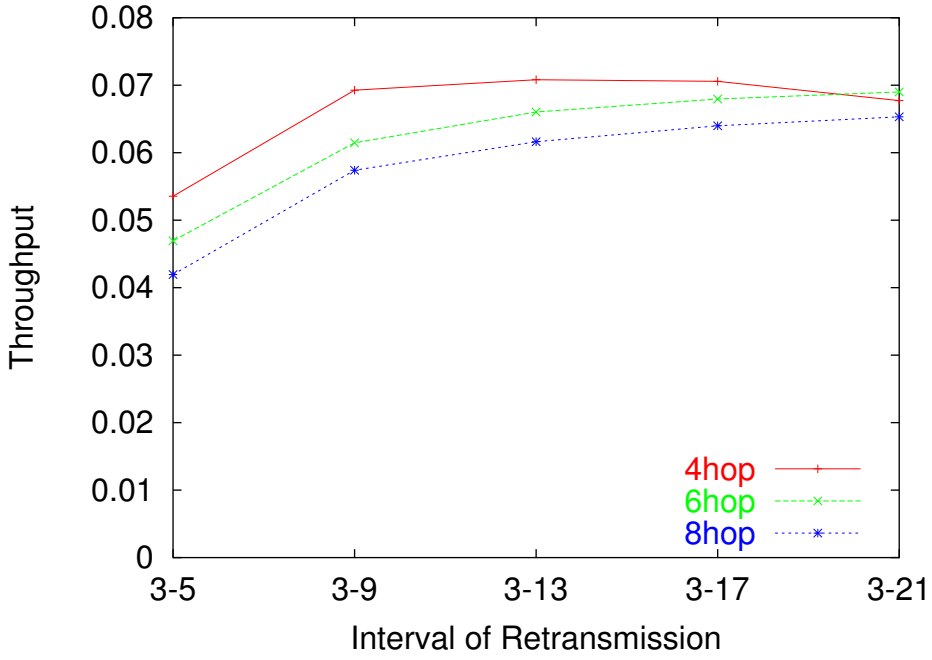


(a) Interval of retransmission: 3-5 time slots

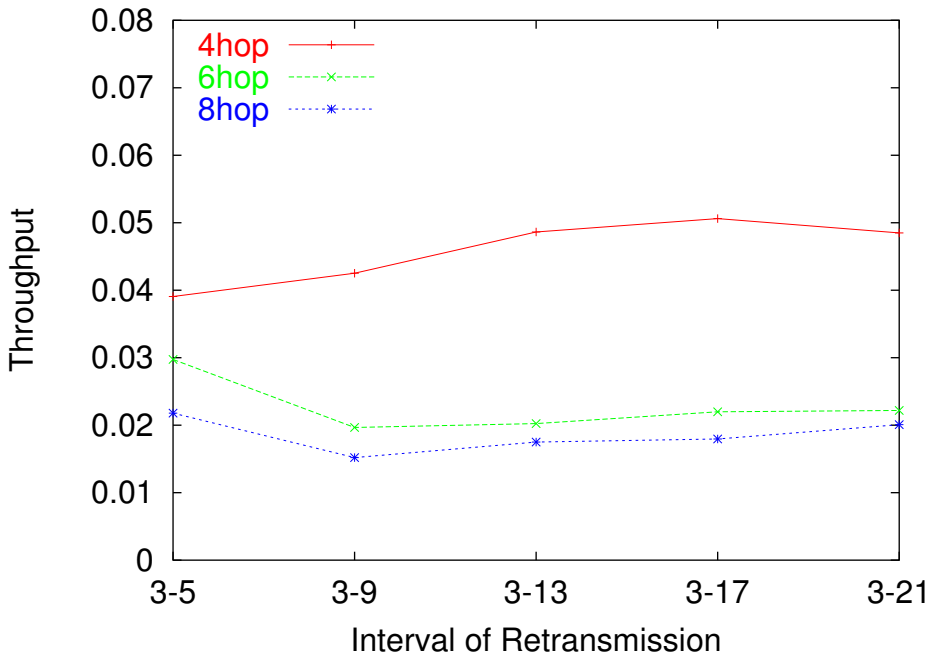


(b) Interval of retransmission: 3-17 time slots

Figure 15: 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%

Figure 16: Change of the throughput according to the retransmission interval

and Figure 18. 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.

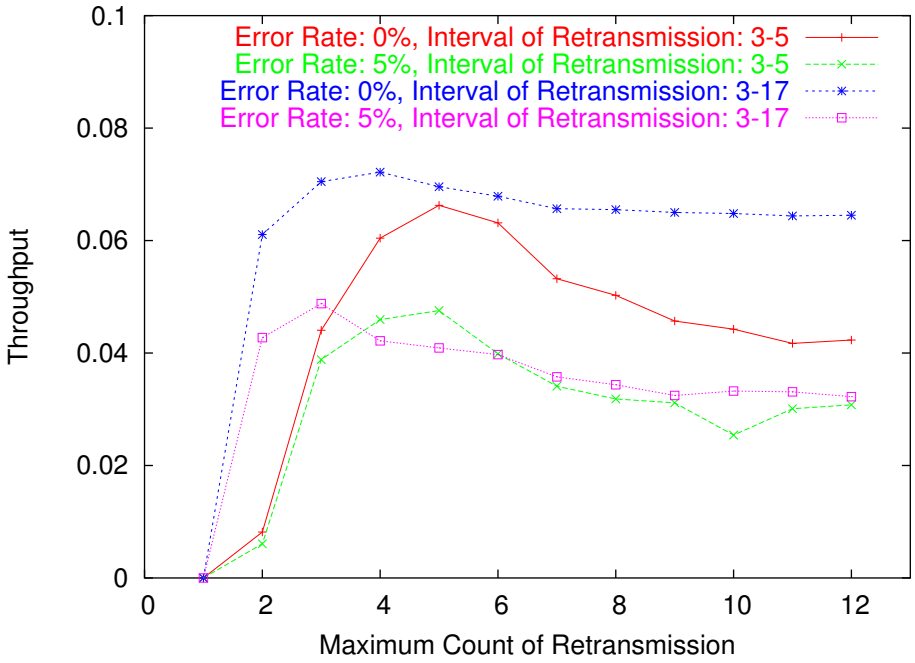
4.3.2.2 Evaluation of Proposed Technique

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.

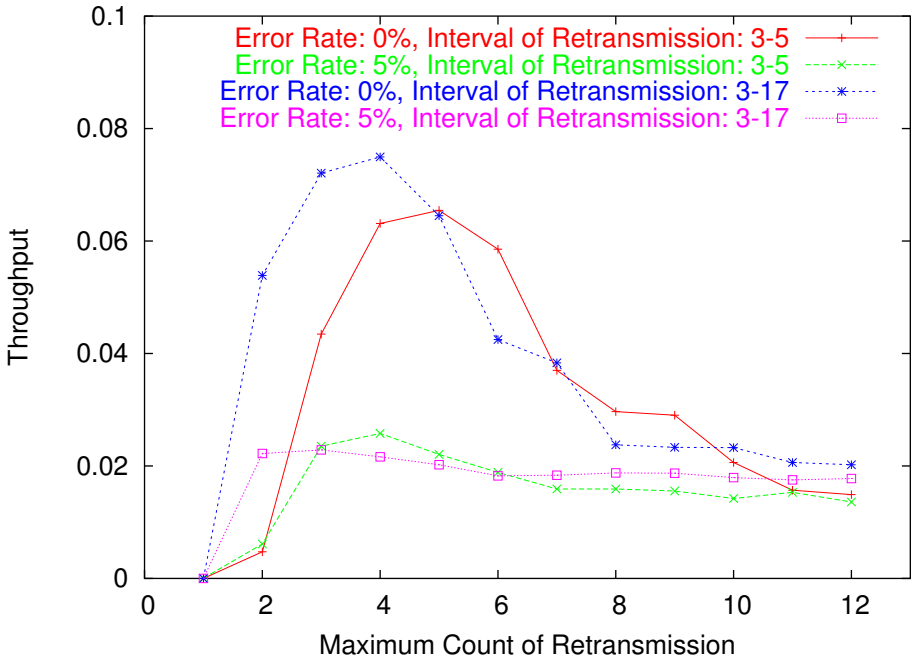
The results are shown in Figure 19. The line of the graph has the the same dispersion of the shortest retransmission interval when judging the load to be nothing. The dispersion of the longest retransmission interval becomes large, so that it travels on a horizontal axis to the right. That is, if the 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 19(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 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 20). 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 19(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) 2 connections of 4 hop in cross-chain topology



(b) 2 connections of 8 hop in cross-chain topology

Figure 17: Change of the throughput according to a maximum count of retransmission in cross-chain topologies

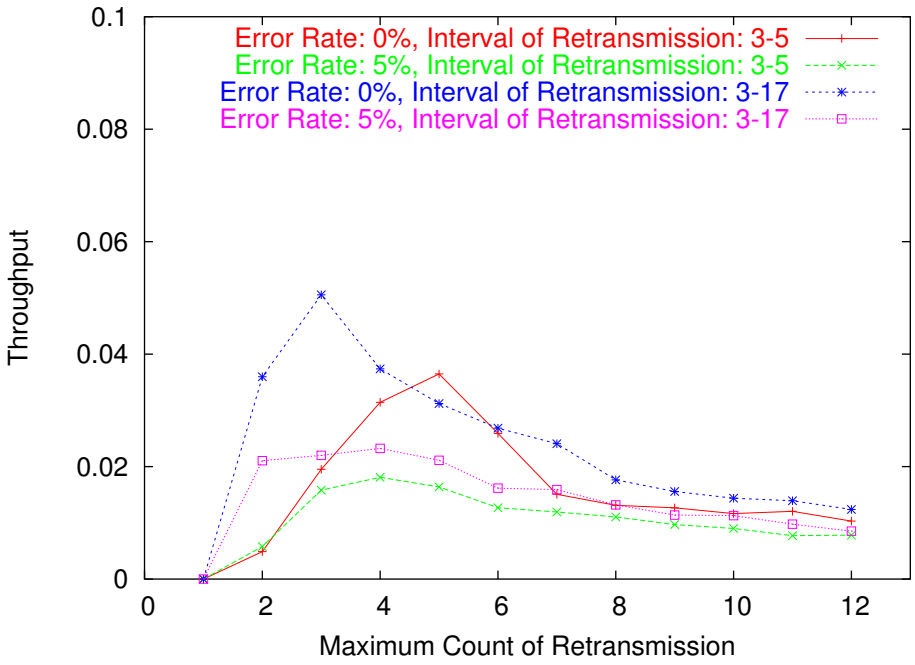
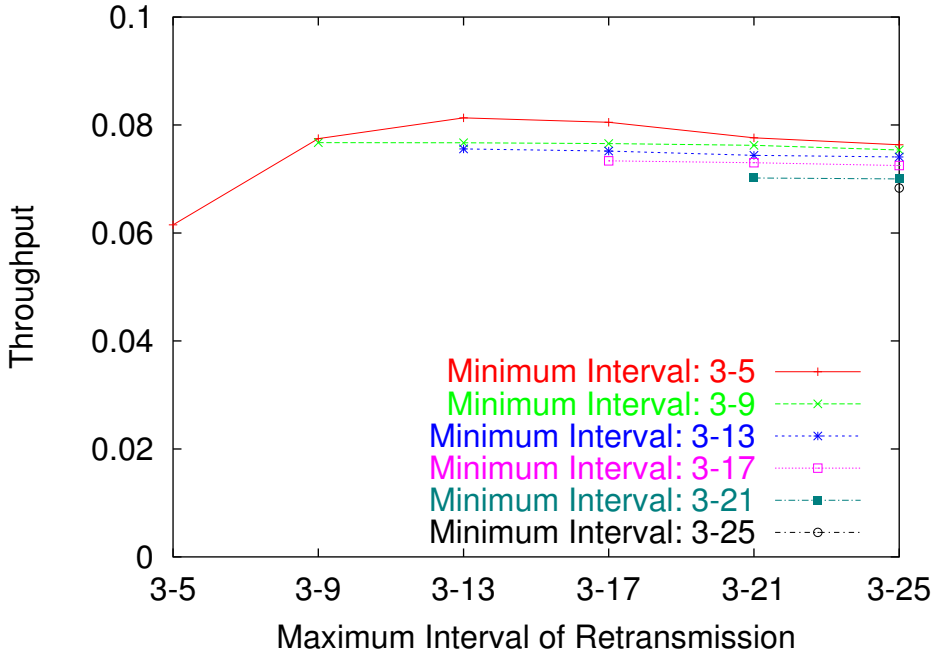
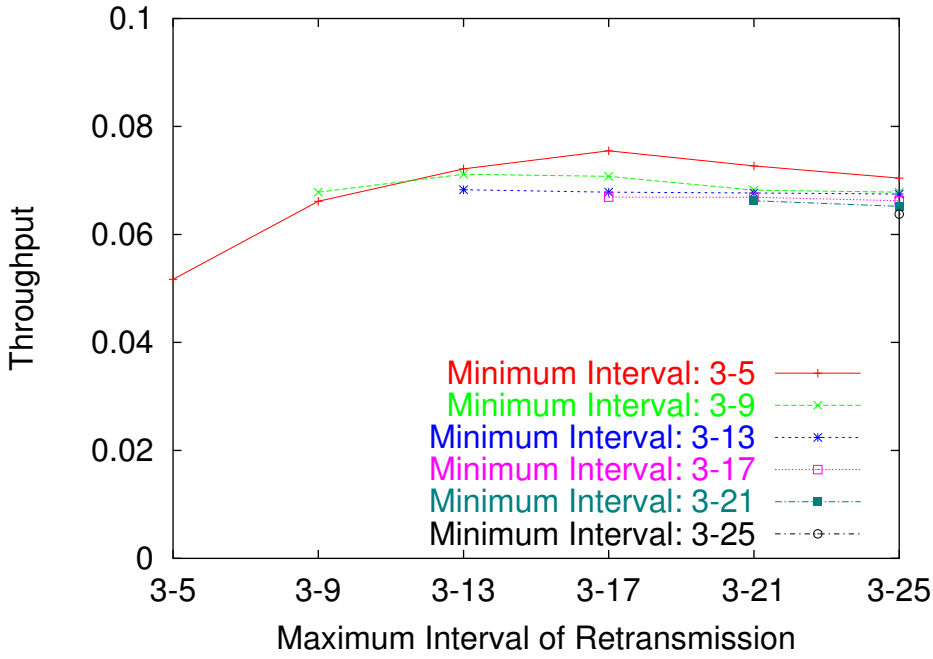


Figure 18: Change of the throughput according to a maximum count of retransmission in mesh topology



(a) 4-hop chain topology



(b) 8-hop chain topology

Figure 19: Change of the throughput when changing a retransmission interval dynamically

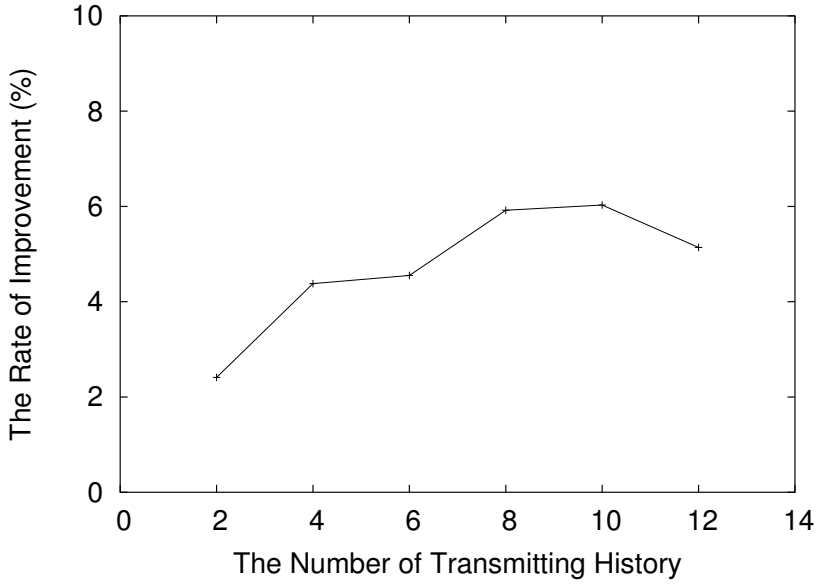
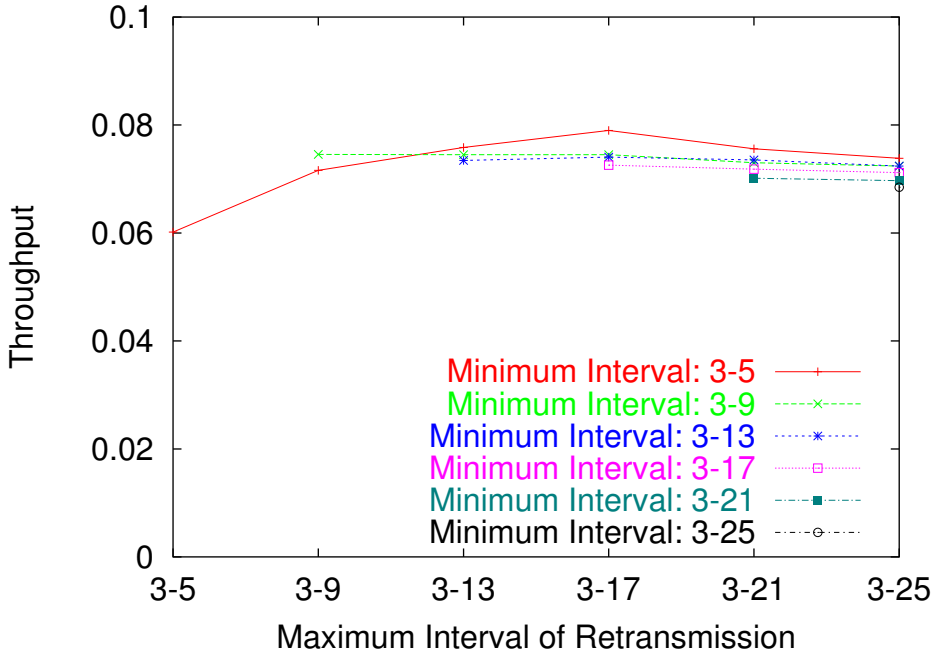


Figure 20: The rate of improvement in each number of transmitting history

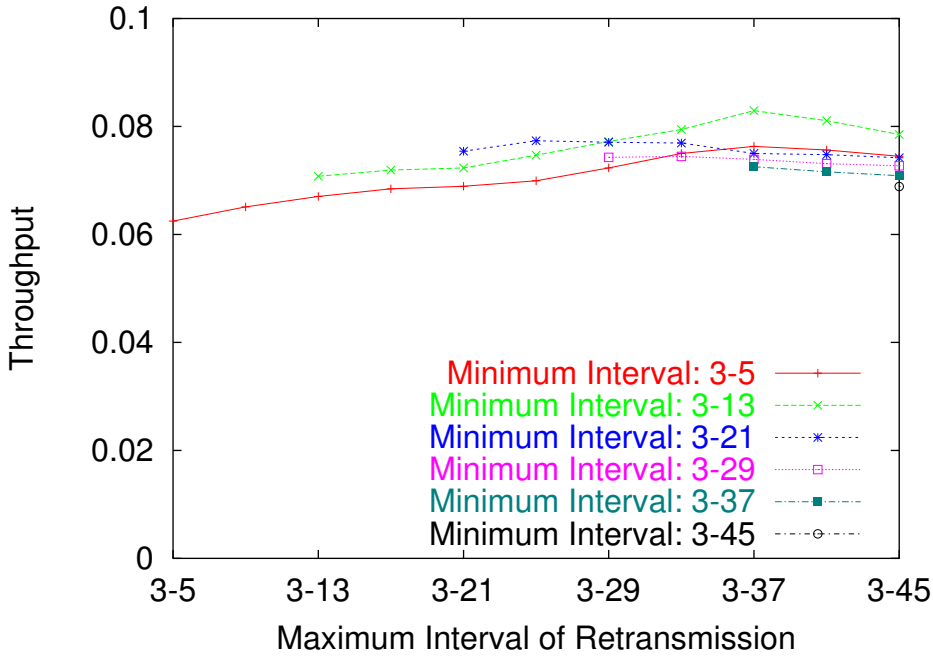
a throughput at the fixed retransmission-interval time of 3-13 time slots. The rate of improvement of the 8 hop is higher than that of the 4 hop. We think that this is because there is a place of high load and low load since the number of the hop is long. Therefore, we look at the effect of the technique by the various topologies.

We evaluate our proposal in a 4-hop cross-chain topology (Figure 5(b)), 8-hop cross-chain topology (Figure 13(c)), and mesh topology (Figure 5(c)). The results are shown in Figures 21 and 22. In the 4-hop cross-chain topology, the tendency changes 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.

Finally, we observe the effect of our proposal when generating the random error of a wireless



(a) 4-hop cross-chain topology



(b) 8-hop cross-chain topology

Figure 21: Change of the throughput when changing a retransmission interval dynamically in cross-chain topologies

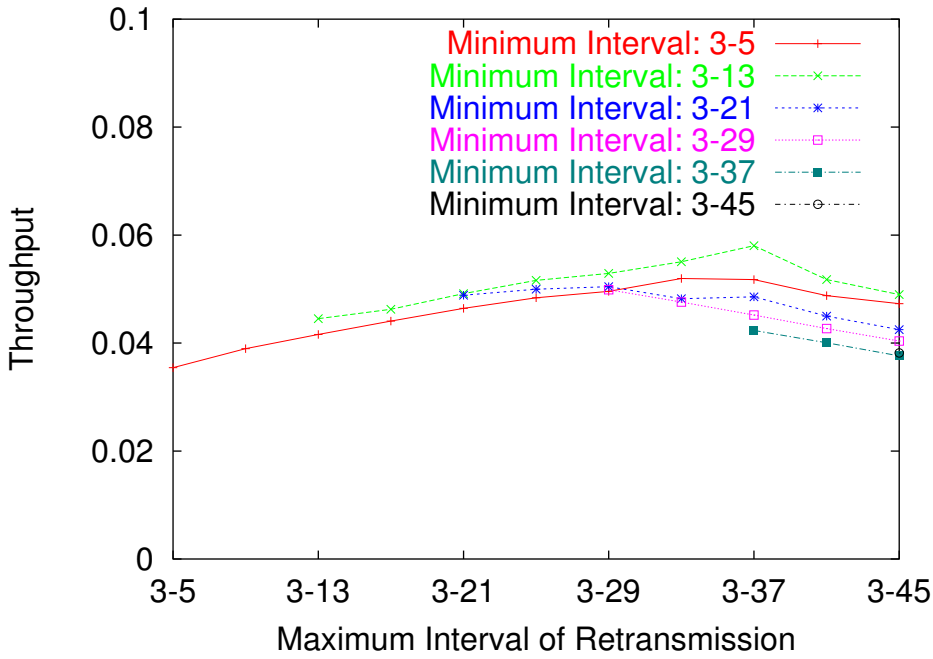
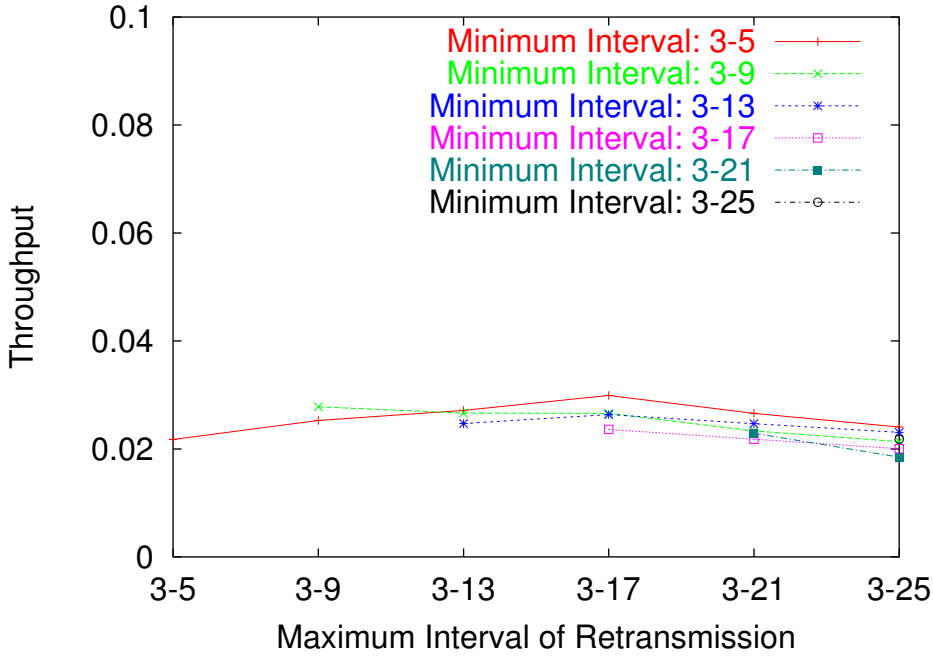
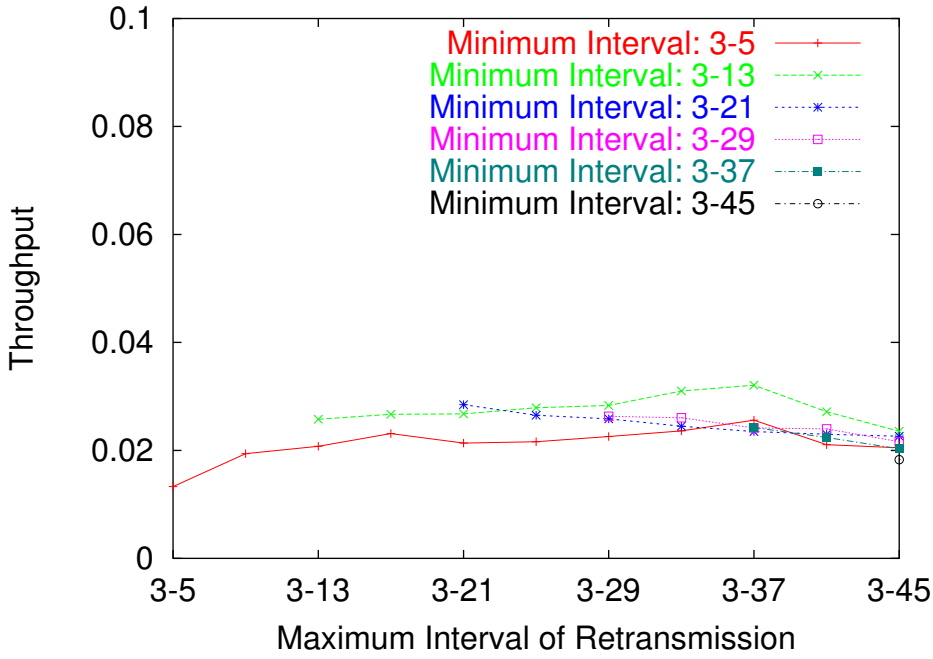


Figure 22: Change of the throughput when changing a retransmission interval dynamically in mesh topology

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 23. 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

Figure 23: Change of the throughput when changing a retransmission interval dynamically with 5% random error of a wireless channel

5 Conclusion

In this thesis, we analyzed TCP problems in an ad hoc network and proposed two techniques for performance improvement from the data link layer.

First, we focused on the problem of packet collisions by the bidirectional communication of TCP. Therefore, we proposed a technique that combines data and ACK packets and demonstrated through simulation that this technique can make wireless 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.

Second, 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.

Acknowledgements

I would like to express my gratitude to Prof. Hideo Miyahara, my supervisor, for his encouragement and valuable comments.

I would like to express my sincere appreciation to Prof. Masayuki Murata for his infinite help and continuous support through my studies.

I am most grateful to Associate Prof. Masashi Sugano of Osaka Prefecture College of Nursing, who has always given me appropriate guidance and invaluable direct advice.

I would like to express my genuine appreciation to Mr. Takayuki Yamamoto, who provided me with helpful comments and advice.

I also deeply appreciate the helpful advice of Mr. Takaaki Hatauchi and Mr. Yohei Hosooka of the Fuji Electric Company.

Without their support, every aspect of this thesis would not have been possible.

I am also indebted to Associate Prof. Naoki Wakamiya, Hiroyuki Ohsaki, and Go Hasegawa of Osaka University for their invaluable advice. I also deeply thank Research Associates Shin'ichi Arakawa and Ichinoshin Maki for their comments and suggestions on the research activities.

Finally, I want to heartily thank my many friends and colleagues in the Department of Information Networking of Osaka University for their support.

References

[1] C. E. Perkins, *Ad Hoc Networking*. Addison-Wesley, 2001.

[2] K. Nakano, M. Sengoku, and S. Shinoda, "Fundamental Characteristics of Multi-hop Wireless Communication Networks," in *Proceedings of 11th ITC Specialist Seminer*, pp. 293–301, Oct. 1998.

[3] C. R. Lin and M. Gerla, "Asynchronous Multimedia Multihop Wireless Networks," in *Proceedings of IEEE INFOCOM '97*, pp. 118–125, Apr. 1997.

[4] A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T. Chen, "Scalable Routing Strategies for Ad Hoc Wireless Networks," *IEEE Journal of Selected Areas in Communications*, vol. 17, pp. 1369–1379, Aug. 1999.

[5] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, "Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks," in *Proceedings of ACM/IEEE MOBICOM '99*, pp. 195–206, Aug. 1999.

[6] D. Kim, C. K. Toh, and Y. Choi, "TCP-BuS: improving TCP performance in wireless ad hoc networks," in *Proceedings of IEEE ICC 2000*, pp. 1707–1713, June 2000.

[7] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," in *Proceedings of ACM/IEEE MOBICOM 2000*, pp. 255–265, Aug. 2000.

[8] N. Nikaein, H. Labiod, and C. Bonnet, "DDR – Distributed dynamic routing algorithm for mobile ad hoc networks," in *Proceedings of ACM MobiHoc 2000*, pp. 19–27, Aug. 2000.

[9] M. R. Pearlman and Z. J. Haas, "Determining the optimal conguration for the zone routing protocol," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 1395–1414, Aug. 1999.

[10] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley, 1994.

[11] 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 ACM/IEEE MOBICOM’99*, pp. 219–230, Aug. 1999.

[12] 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 IEEE GLOBECOM 2002*, Nov. 2002.

[13] 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 *Proceedings of IEEE INFOCOM 2003*, Mar. 2003.

[14] 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, pp. 34–39, Feb. 2001.

[15] 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.

[16] Z. Fu, B. Greenstein, X. Meng, and S. Lu, “Design and implementation of a TCP-friendly transport protocol for ad hoc wireless networks,” in *Proceedings of IEEE ICNP’02*, pp. 216–225, Nov. 2002.

[17] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar, “ATP: a reliable transport protocol for ad hoc networks,” in *Proceedings of ACM MobiHoc 2003*, pp. 64–75, June 2003.

[18] 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.

[19] S. Ray, J. Carruthers, and D. Staorobinski, “RTS/CTS-Induced congestion in ad hoc wireless LANs,” in *Proceedings of IEEE WCNC’03*, pp. 1516–1521, Mar. 2003.

[20] “Flexible Radio Network, Fuji Electric Co. Ltd.” available at http://www.fujielectric.co.jp/denki/p26/ecop_contents2.html.

[21] M. Sugano, T. Araki, M. Murata, T. Hatauchi, and Y. Hosooka, “Performance evaluation of a wireless ad hoc network: Flexible radio network (FRN),” in *Proceedings of IEEE ICPWC 2000*, pp. 350–354, Dec. 2000.

[22] “The Network Simulator – ns–2.” available at http://www.isi.edu/nsnam/ns/.

[23] “The CMU monarch project.” available at http://www.monarch.cs.cmu.edu/.