Regular Paper

A Distributed Scheduling through Queue-length Exchange in CSMA-based Wireless Mesh Networks

Toshiki Takeda[1] Takuya Yoshihiro[2,a)]

Received: March 29, 2016, Accepted: November 1, 2016

Abstract: Wireless Mesh Networks (WMNs) over CSMA MAC (especially IEEE 802.11) are an attractive solution to widen the coverage area of the Internet in unlicensed frequency bands. Although such CSMA-based WMNs have been deeply investigated for a long time, they still suffer from heavy interference due to hidden terminals. In this paper, we accelerate the performance of CSMA-based WMNs by introducing a distributed scheduling scheme that exchanges the transmission-queue length information in real-time among neighbor nodes. In our scheduling scheme, we exchange the information of transmission queue-lengths among neighbor nodes in real-time, and allow transmitting frames to the node that has the longest queue length among its 2-hop distance. The proposed scheduling scheme can be regarded as a distributed design of so called 'Max-weight' scheduling. By combining CSMA with the queue-length based scheduling, we significantly reduce collisions due to hidden terminals and improve the performance with a small overhead of queue-length fields in MAC frames.

Keywords: Wireless Mesh Network, IEEE802.11, MAC protocols, and Queue-length

1. Introduction

Wireless Mesh Networks (WMNs) using commodity IEEE802.11 network interfaces have been extensively studied for a long time to achieve low-cost wireless network infrastructures [1]. However, such CSMA-based WMNs still suffer from heavy collision due to hidden terminals that significantly degrade the communication performance. In fact, the effect of hidden terminals leads to not only low communication speed but also poor packet delivery ratio and delay. Due to the unwanted characteristics, CSMA-based WMNs (with populated omni-directional antennas) are rarely used in practical scenes at present.

To improve the communication performance, considerable efforts have been dedicated so far. However, no promising mechanism has appeared to solve the collision problem due to hidden terminals. The best-known mechanism would be RTS/CTS handshake that is adopted in IEEE802.11 standards [2], [3], which makes a time reservation for transmitting a long data frame by exchanging short RTS/CTS messages before transmitting a long data frame. RTS/CTS actually improves the communication performance especially in high-traffic cases with long data frames. However, in the context of WMNs, it is known that RTS/CTS yields limited performance due to collision of RTS/CTS themselves, or due to so called exposed terminal problems, etc. [4]. In addition, Xu et al. reported that the difference between the communication range and the interference range causes collisions even under presence of RTS/CTS [5]. These studies show that the effect of RTS/CTS against hidden terminals is quite limited in WMNs.

In order to improve the communication performance of CSMA-based WMNs up to a practical level, a wide variety of approaches have been presented so far. A part of major approaches appears in a network layer. For example, several multi-path routing protocols for load balancing have appeared [6]. Also, to compute high-quality paths in routing protocols, several link metrics used in the shortest-path computation have been proposed such as ETX [7], ETT [8], etc. However, because they essentially do not solve collisions, frames inevitably collide due to hidden terminals when several communication paths cross one another. As a result, from the network layer approaches, the effect to reduce collision is limited.

Multi-channel utilization is also a major approach to reduce collisions. Several static channel assignment algorithms such as CLICA [9] that minimize the number of conflicting pairs of links have been studied. However, because IEEE802.11 has too small a number of orthogonal frequency channels (consider that 2.4 GHz band has only 3 orthogonal channels), it is essentially impossible to sufficiently reduce the conflicts coming from hidden terminals. Also, the number of channels available for WMNs is often limited for practical reasons.

To overcome the shortage of frequency channels, several dynamic multi-channel utilization methods for WMNs have been proposed. For example, Raniwala et al. proposed a dynamic channel utilization mechanism in which each interface adaptively switches its assigned channels [10]. Draves et al. considered a network in which all nodes have k network interfaces assigned with the same set of frequency channels, and the best one is selected by routing decision [8]. Kanaoka et al. also tried to achieve

[1] Graduate School of Systems Engineering, Wakayama University, Wakayama 640–8510, Japan
[2] Faculty of Systems Engineering, Wakayama University, Wakayama 640–8510, Japan
[a)] tac@sys.wakayama-u.ac.jp

more efficient usage of frequency channels in the same environment [11]. However, the essential problem of channel shortage is still unsolved so that significant collisions still remain that prevent WMNs from achieving practical performance.

As shown above, an essential solution should be made to prevent hidden-terminal problem as a MAC technique in a framework of single channel CSMA-based WMNs. However, to the best of our knowledge, no such promising technique has been presented yet. Recently, Sheshadri et al. tried to evaluate WMNs based on IEEE802.11n, which also results in poor performance due to severe interference [12].

In this paper, we propose a new distributed scheduling mechanism to significantly reduce the conflicts due to hidden terminals within a framework of single-channel CSMA-based WMNs. The key idea is to share the output-queue length of each node within a two-hop neighborhood via data and Ack frames of CSMA, and accordingly we control transmission timing to avoid simultaneous transmission among two-hop (i.e., hidden-terminal) nodes. By utilizing data and Ack frames, nodes are able to watch their output-queue lengths one another in quasi-real-time and thus able to switch 'active' nodes that are allowed to transmit data frames quickly and properly. Through evaluation, we clarified that our scheduling scheme significantly reduces collisions due to hidden terminals in a general CSMA-based WMNs. Although our scheme has overhead that data and Ack frames must include queue-length information, the benefit in terms of communication performance is large and clearly exceeds the overhead.

This paper is organized as follows: In Section 2 we introduce several technical topics related to our method. In Section 3 we describe the proposed method in detail. In Section 4 we report the evaluation results, and we conclude the work in Section 5.

2. Related Work

Several aspects of researches on WMNs are related to this work. Although we describe the related work on CSMA-based WMNs in the previous section, several other topics are still worth noting.

Firstly we refer to the queue-length based scheduling. In the context of time-slotted architecture such as TDMA, Tassiulas et al. presented a queue-length based scheduling called 'Max-Weight' in which the link with the longest output queue among all conflicting links obtains the slot for transmission [13]. Although this strategy is difficult to implement due to computational complexity, this scheduling has been theoretically proven to be optimal, i.e., it is proven that any communication requirement that has a schedule that satisfy it can also be satisfied by 'Max-Weight' schedule. Thus, many studies followed this result to achieve sub-optimal scheduling schemes based on queue-length [14]. Among the following studies, Li et al. tried to introduce queue-length-based strategy into CSMA-based WMNs [15]. However, because they only control the back-off time for contention according to output-queue length, essentially collisions are not solved practically. The contribution of this paper is proposing a new distributed scheduling algorithms inspired by 'Max-Weight' optimal scheduling that works in combination with CSMA such as IEEE 802.11 and significantly reduce the harmful effect of hidden terminals.

Secondly, we note that there are several distributed scheduling schemes proposed for TDMA context. Bao, et al. proposed a neighborhood contention resolution (NCR) technique for time-slotted systems that exchanges IDs within the two-hop neighborhood and determines the contention winner using a pseudo-random function [16]. TRAMA [18] has a scheduling random access period and a scheduled period where distributed scheduling is done in the random access period so as not to have any pair of nodes within 2-hop distance transmits a frame simultaneously. DRAND [19] that is designed to work in combination with Z-MAC is also a distributed scheduling in the context of TDMA in which no 2-hop neighbors are assigned to the same slot. However, they are basically designed for time-slotted schemes such as TDMA. Our proposal is the first trial to incorporate a distributed scheduling methodology into fully CSMA-based WMNs in order to eliminate the hidden terminal problem.

Finally, as a MAC protocol that incorporates scheduling within CSMA, Wu, et al. [20] proposed Distributed Link Scheduling Multiple Access Protocol (D-LSMA). Their protocol divides the time axis into consecutive time slices, and reserves periodic time slots within the time slices using RTS/CTS. This reservation mechanism using RTS/CTS makes successively transmitted real-time communications collision-free with a single RTS/CTS handshake. They also succeeded in removing exposed-terminal effects considering that real-time traffic does not require Ack frames to invoke retransmissions. However, D-LSMA is different in scope from ours in that they only assume real-time traffic such as video or sound streaming.

3. Queue-length Based Distributed Scheduling

3.1 Overview

We assume a CSMA-based WMN where each node equips a single network interface with an omni-directional antenna assigned to a single common frequency channel. In this situation, hidden terminals invoke severe collision among MAC frames, which terribly degrades the communication performance. To avoid collision within the framework of CSMA, we introduce a scheduling scheme in which a node takes two states, *active* and *inactive*; an *active* node behaves the same as CSMA and an *inactive* node does the same except that they do not transmit data frames. Note that *inactive* nodes can transmit Ack frames so that transmissions from an *active* node to an *inactive* node will succeed. Because the modification is small, our scheme can be implemented easily in the commodity IEEE802.11 interfaces.

To control the state of nodes, we use the information of output-queue lengths within two-hop neighborhood. Namely, we also assume that each node has a single output queue. Note that this is different from the assumption of 'Max-weight' scheduling [13]. In this paper, we try to apply the 'node-scheduling' that activate/inactivate nodes into CSMA-based WMNs, instead of 'link-scheduling' that activate/inactivate links as is done in 'Max-weight' scheduling, since the commodity IEEE802.11 devices usually have a single output queue.

More specifically, we send a message that includes queue-lengths information piggy-backed by a data and Ack frame. The

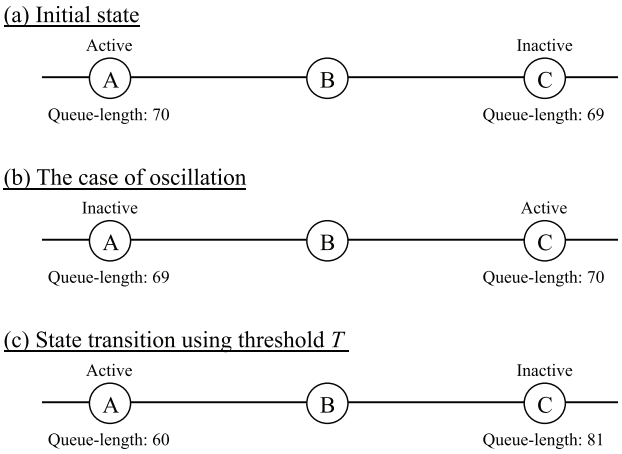


Fig. 1 State transition of nodes.

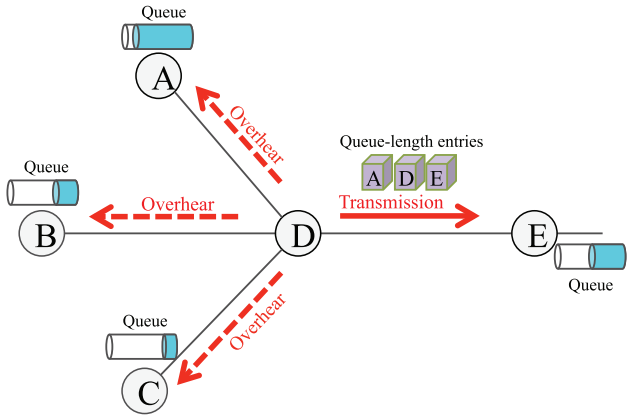


Fig. 2 Message format for Queue-length notification.

receiver of the frame as well as other overhearing nodes obtain the information. Because the queue length changes only when a data frame is sent or received, we can expect that every node is mostly able to watch the queue lengths of neighborhood nodes with very low delay.

The state of a node is basically determined according to the queue lengths of the node itself and its two-hop neighbors. When a node received or overheard a message from another node, it makes a decision to change its state. The basic strategy in our scheduling is that only the node that has the longest queue length among its two-hop neighbors becomes *active*, and others remain *inactive*. However, we have to add a simple technique to avoid the problem of state oscillation.

See **Fig. 1** (a). Suppose that nodes A and C are in two-hop distance, and their queue-lengths are 70 and 69, respectively. If the queue-length rule is applied strictly, when one packet moves from A to C, the state should be changed as shown in Fig. 1 (b) where both states of A and C are changed. Now because C is active, C in turn transmits packets from its queue, and soon the state would again return to Fig. 1 (a). This oscillation may continue repeatedly. Since the state change includes a little overhead such as the risk for collision, this state oscillation seriously degrades the performance.

Thus, we additionally introduce a threshold value T_{seesaw} to slow down the oscillation into 'seesaw'-like behavior. See Fig. 1 again. If A sends out several frames and C receives several frames on the situation of Fig. 1 (a), the situation will be like Fig. 1 (c) in which the queue-lengths of A and C are 60 and 81, respectively. Assume that we set the threshold $T_{seesaw} = 20$, and now the queue length of C is larger than A by more than T_{seesaw}, then the states of A and C change to *inactive* and *active*, respectively. In this way, by determining the threshold T_{seesaw} to an appropriate value, we control the oscillation speed so as to provide a good communication performance.

From a network-wide view, in our scheduling, nodes that have the largest queue-lengths among their two-hop neighbors become *active*. Each time a node become *active*, it transmits a small collection of frames, and then passes its *active* state onto the next node, i.e., one of its 2-hop nodes. By transmitting a collection of frames by turns, our scheduling achieves efficient and low-collision communications.

3.2 Message Format

In our scheduling scheme, queue length information is propagated to two-hop neighborhood as a message included in a data or Ack frame. In our design, the size of messages is constant, and each message consists of three queue length entries. A queue length entry is a triplet of values; a node ID, a queue-length, and a current state of the corresponding node. To make the most of these three entries, each entry corresponds to the following nodes:

(i) the transmitting node itself,
(ii) the destination node of the frame,
(iii) and the largest queue length node among all neighbors except for the destination node (ii).

Figure 2 shows the example of these three entries. Suppose that node D transmits a frame destined to E. In this case, the message in the frame includes the entries of three nodes D ((i) transmitting node), E ((ii) destination node), and A ((iii) the largest queue length node among A, B, and C). With these three entries (i) (ii) and (iii), we can propagate information as follows: (i) D informs all neighbors of the queue length of D itself, (ii) E obtains the information of the largest queue length 2-hop node in the direction of D, and (iii) A, B and C obtains the information of 2-hop node E. In this way, a message from a node enables its neighbors to exchange the information of 2-hop neighbors efficiently.

Note that the encoding method or format of a queue length entry is not specified in this paper. However, for reference, we present an example of encoding that is used in our evaluation as follows:

Node ID: for node ID, we use the number represented by 7 bits. Since the requirement for node ID in our scheduling is to distinguish nodes within two-hop neighborhoods, 7 bits would be sufficient. We can assign IDs manually, or we can design a protocol to assign IDs to nodes that are unique within two-hop neighborhood of any node.

Current state: naturally, the current state of a node, i.e., *active* or *inactive*, can be represented by 1 bit.

Queue length: for queue length, we use 8 bits. We may encode the real queue length linearly to 8 bit values, but in this paper we use logarithm encoding. Namely, we take the logarithm of the real queue length, and encode it linearly to the 8-bit values,

shown as the following:

$$L = \mathrm{ceil}\left(\frac{\log(Q_{len}+1)}{\log(Q_{\max}+1)} \times 254\right), \quad (1)$$

where L is the encoded 8-bit values, $ceil(\cdot)$ is the ceiling function that returns the interger value, Q_{len} is the real queue length, and $Q_{\max}$ is the maximum possible length of the output queue. With this logarithm-based encoding, the seesaw behavior slows down as the queue length becomes longer, and thus the overhead in changing states become lower. This encoding supports more efficient communication in higher-traffic scenarios.

3.3 Message Propagation

As described above, queue length information is sent as a message included in MAC frames. In this section we watch the process of propagating messages and clarify how every node gets to know the queue length information of every two-hop neighbors.

We first begin with the process to obtain the information of neighbors. Recall that a message includes the entry of the transmitter itself (i.e., entry (i) in the previous section). Because neighbors can receive or overhear the data or Ack frames every time the queue length changes, nodes always know the queue length of their neighbors. Note that collisions may occur, and which causes failure to receive messages, but we consider that the probability of the collision is sufficiently low.

In propagating queue length information of two-hop neighbors, we consider four cases according to the state of nodes shown in **Fig. 3**. There are three nodes A, B and C, and we assume a flow that travels from A to C exists.

In case (a) we assume that A and B are both *active* state, and packets travel along $A \rightarrow B \rightarrow C$. Note that, when B transmits a frame, A and C hear it, while B knows the accurate queue-length of A and C. As a result, nodes in two-hop distance A and C get to know the queue length of each other via B. Namely, A and C can watch each other's queue length in quasi-real-time.

In case (b) where only A is *Active*, the result is the same. When A transmits a data frame, B returns an Ack frame that can be overheard by both A and C. Since B knows the accurate queue lengths of both A and C, A and C are able to know the information of each other.

Case (c) also results in the same as cases (a) and (b) because B is active, i.e., even if A is *Inactive*, B knows the queue-length of A in quasi-real-time, and thus when B transmits a frame, A and C are able to know the information of each other.

Only case (d) is inconvenient for us, where A and C may know queue lengths for each other that are wrong because no message may be received even when the queue lengths of A or C are updated. This pattern may invoke inconsistency among node states that may cause a drop in communication performance.

3.4 The State Transition Rules

In our scheme, nodes behave autonomously according to the protocols that we present in the following. When a node receives a frame, it stores the included message into its database, and changes its state according to the rule set. The rule set is shown in **Table 1**, where the conditions and their corresponding 'actions' that nodes take are shown. Here, Q_{self} represents the queue length of the node itself, Q_{act} does the maximum queue length among *active* two-hop nodes, Q_{inact} among *inactive* two-hop nodes. The seesaw-like state transition is implemented in this rule set. As shown in Table 1, the actions to change the node state are determined according to the state of the node itself, states of two-hop nodes, and the conditions on queue-lengths.

However, the policy of the rule set is simple; each node simply compares the queue length with their two-hop nodes, and if the node has the longest queue, it becomes *active*, and otherwise, it

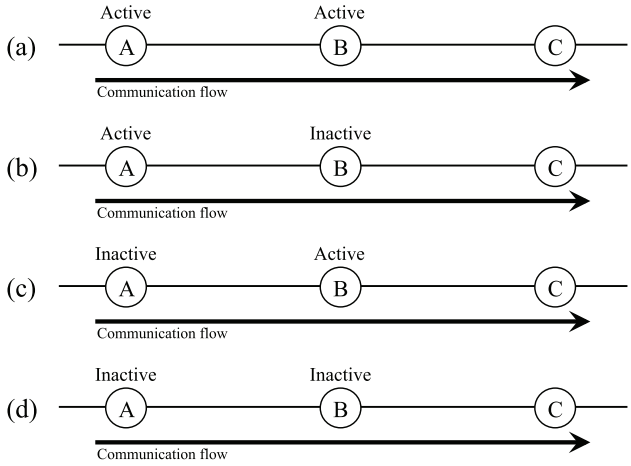


Fig. 3 Four patterns of Queue length propagation.

Table 1 Formal description of state transition rules.

State of the Node	States of Two-hop Nodes	Conditions	Action
Active	At least one *Active* node exists	$Q_{self} > Q_{act}$ and $Q_{self} > Q_{inact} - T_{seesaw}$	Remain *Active*
		$Q_{self} \leq Q_{act}$ or $Q_{self} \leq Q_{inact} - T_{seesaw}$	Transit to *Inactive*
	All nodes are *Inactive*	$Q_{self} > Q_{inact} - T_{seesaw}$	Remain *Active*
		$Q_{self} \leq Q_{inact} - T_{seesaw}$	Transit to *Inactive*
Inactive	At least one *Active* node exists	$Q_{self} \geq Q_{act} + T_{seesaw}$ and $Q_{self} \geq Q_{inact}$	Transit to *Active*
		$Q_{self} < Q_{act} + T_{seesaw}$ or $Q_{self} < Q_{inact}$	Remain *Inactive*
	All nodes are *Inactive*	$Q_{self} \geq Q_{inact}$	Transit to *Active*
		$Q_{self} < Q_{inact}$	Remain *Inactive*

becomes *inactive*. Here, we apply the threshold T_{seesaw} if *active* and *inactive* nodes are compared, and we do not apply it when we compare other patterns of two nodes.

We have to consider several points in designing the rule set. First, we have to prepare the rules for the cases where there are inconsistencies in the state of nodes. Actually, we frequently experience the case where two nodes in two-hop distance are both *active*, or the case where there is no *active* node in the two-hop neighbors of an *inactive* node. These cases occur when there is something wrong with the message propagation process, caused by collisions, etc. Actions for this kind of inconsistencies are implemented in the rule set.

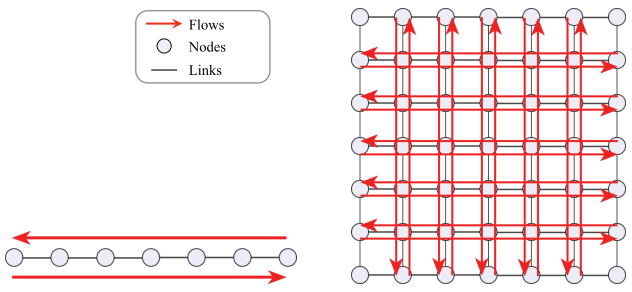
Second, from the similar reason, we have to introduce expiration time of messages. When some inconsistency occurs, nodes may fall into a kind of 'deadlock' in which states would not be proceed further. To avoid this situation, every node records the time that each massage is received, and if it is not updated until the expiration time $T_{timeout}$ passes, the node silently deletes the message from its database.

Third, we note that the proposed scheduling scheme effectively works only in high-traffic networks. In contrast, in low-traffic cases, although CSMA is sufficient to cope with the traffic requirement, the proposed scheduling may yield unfavorable results; when the queue-length changes slowly, seesaw behavior would increase end-to-end delay of packets. Thus, we should introduce a technique to prevent this problem. For example, we would propose to introduce a threshold $T_{\min}$ and nodes never become *inactive* state unless at least one queue length of two-hop neighbors exceeds $T_{\min}$. By limiting the proposed method work only in high-traffic situations, we can avoid this kind of inconvenience.

4. Evaluation

4.1 Methods

We evaluate the proposed scheduling mechanisms through simulation. As a network simulator, we use Scenargie [21] that implements up-to-date physical layer models as well as IEEE802.11 MAC layer models. We tested two network topologies. One is the line topology shown in **Fig. 4** (a) to measure the basic performance of the proposed method, in which two bi-directional flows are generated. To measure the practical performance in more general networks, we prepare the other topology, 7×7 grid topology, in which bi-directional flows between 10 pairs



(a) Line topology (b) Grid topology

Fig. 4 Evaluation scenarios.

of nodes are generated. The interval between nodes is 300 meters so that 15 dB transmission power reaches to only the neighbors in the vertical or horizontal direction.

Flows we generated are all CBR (Constant Bit Rate) flows that transmit 512-bytes UDP datagrams in a common transmission rate. Routing tables are statically configured to use the shortest path to destinations. We use IEEE802.11g with 6 Mbps (BPSK0.5) standard as PHY and MAC protocols over two-ray-ground propagation model. The simulation time is 6 minutes and we use the average of 5 repetitions.

In our evaluation, we compare the performance of the proposed method with IEEE802.11g, which is a major standard that adopts CSMA, with and without RTS/CTS handshakes. Note that we in this paper propose a MAC protocol that accelerates IEEE 802.11 without modifying the standard. Thus, RTS/CTS would be the best one to be compared among the MAC mechanisms that work in the framework of CSMA. As evaluation criterion, we use aggregated throughput, delivery ratio, and the number of frame loss due to collision over the network. As for the parameter values $T_{timeout}$ and T_{seesaw}, we use the values $T_{timeout} = 50$ msec and $T_{seesaw} = 26$, which are the best performance values in our preliminary test shown in the next section.

4.2 Parameter Tuning

We firstly determine the parameters $T_{timeout}$ and T_{seesaw} through preliminary simulations. As a simulation scenario, we use 7×7 grid topology with 20 directional flows shown in Fig. 4 (b).

To see the best value of $T_{timeout}$, we vary $T_{timeout}$ from 10 to 100 msec while the other values are fixed as $T_{seesaw} = 26$, and transmission rate 128 Kbps (i.e., 2.56 Mbps in total). The result is shown in **Fig. 5**, which shows that there is a range of $t_{timeout}$, i.e., 30–60 msec, in which the proposed method performs well.

Next, we vary T_{seesaw} from 1 to 26 with $T_{timeout} = 50$ msec. The transmission rate of flows is the same as the previous simulation, i.e., 128 Kbps. The result is shown in **Fig. 6**, which shows that the large t_{seesaw} performs well. Also, we see that the difference in the performance goes smaller as T_{seesaw} goes large, e.g., the difference between $T_{seesaw} = 20$ and 26 is small. Thus, we conclude that the value of T_{seesaw} larger than 20 performs well.

4.3 Main Results

The results of the line topology are shown in Figs. 7–10. **Figure 7** shows the aggregated throughput of each method in which the proposed method outperforms both CSMA with and without RTS/CTS. This means that the benefit of the proposed scheduling is far larger than the overhead of queue-length messages included in data and Ack frames. Note that RTS/CTS performs better than CSMA, which is because packet size (512 bytes) is sufficiently large for RTS/CTS to reduce collision. Nevertheless, the proposed method achieves far better collision reduction effects. **Figure 8** shows the ratio of packets that reaches its destination. The proposed method also outperforms the others, and maintains almost 100% delivery with larger transmission rate. **Figure 9** shows the number of frames that are lost due to collision among frames. The proposed method again has the best performance, where the dropped frames take similar values after the transmis-

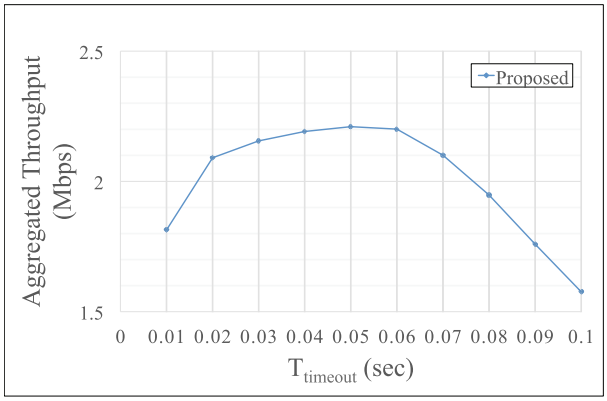


Fig. 5 Throughput under variation of $t_{timeout}$.

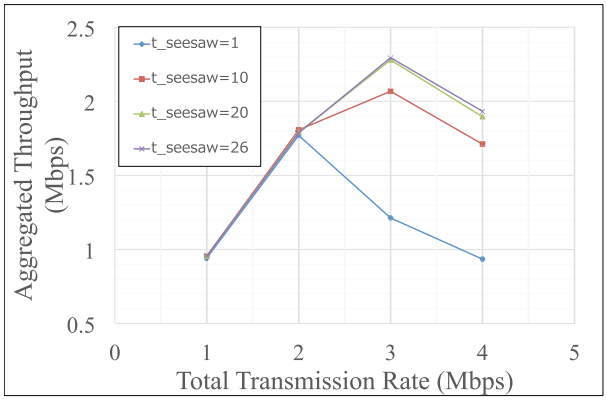


Fig. 6 Throughput under variation of t_{seesaw}.

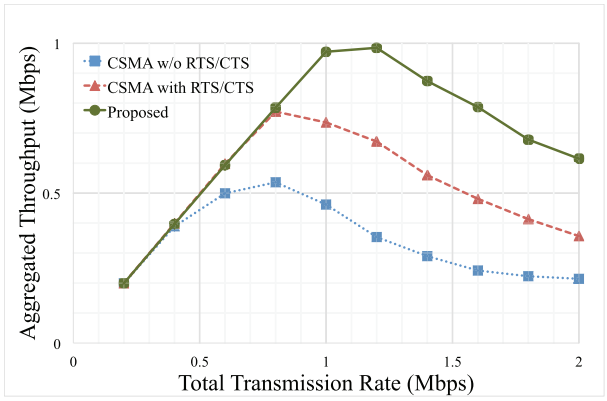


Fig. 7 Throughput (line topology).

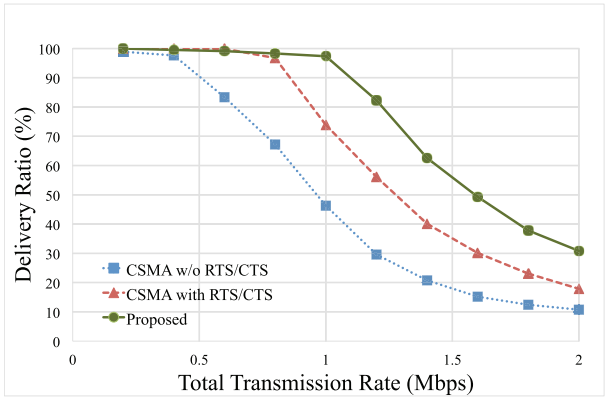


Fig. 8 Delivery ratio (line topology).

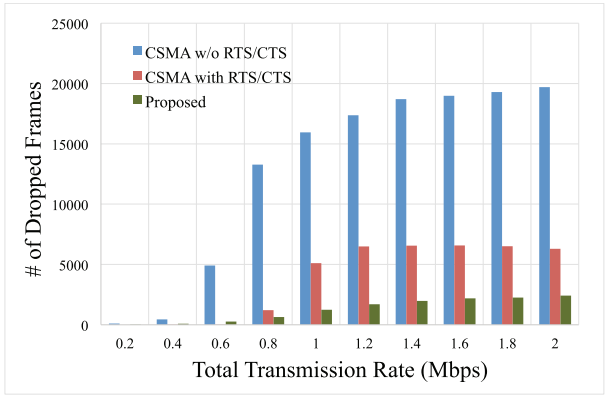


Fig. 9 Dropped frames (line topology).

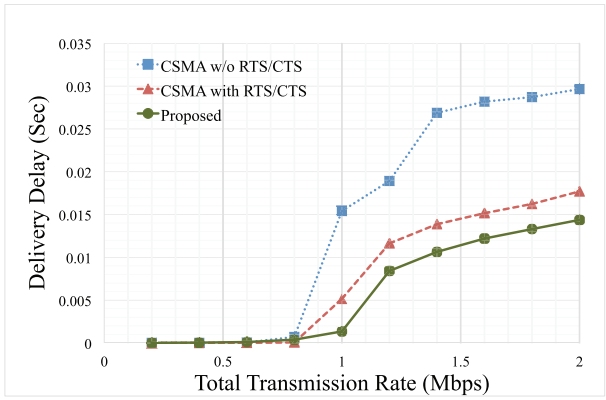


Fig. 10 Delivery delay (line topology).

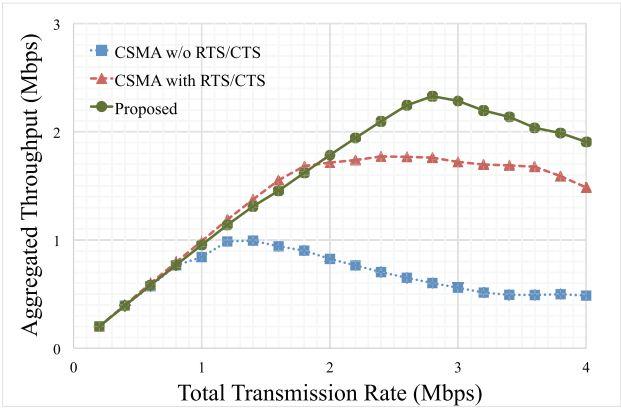


Fig. 11 Throughput (grid topology).

sion rate exceeds 1.2 Mbps. Note that over 1.2 Mbps we have severe queue-over drops, meaning that the proposed method is able to keep a low collision level even in the heaviest collided situation. **Figure 10** compares the delivery delay, which also shows that the proposed method outperforms the others.

On the other hand, Figs. 11–14 shows the results of the grid topology. Also in the grid topology, the proposed method outperformed the others from the viewpoint of throughput (**Fig. 11**), delivery ratio (**Fig. 12**), and dropped frames (**Fig. 13**). However, we would note that the packet delivery ratio of the proposed method failed to keep around 100% and gradually decreases even in case of low transmission rate. This is because in the grid topology two-hop nodes (i.e., hidden terminals) increase and accordingly message propagation failure increases. In the log file, we found

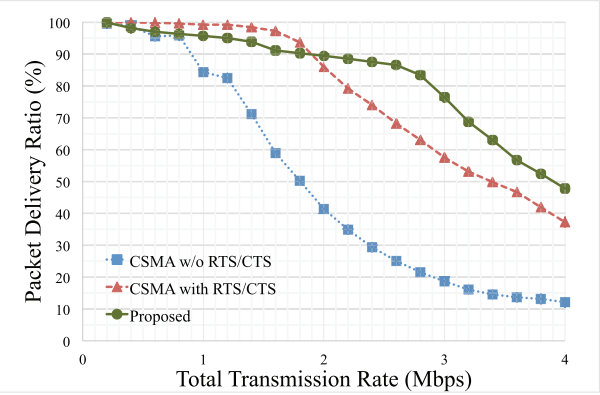


Fig. 12 Delivery ratio (grid topology).

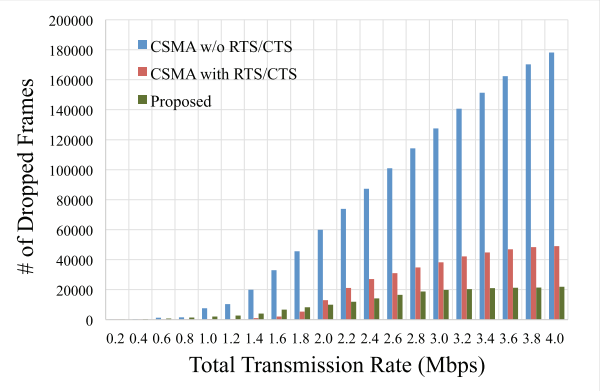


Fig. 13 Dropped frames (grid topology).

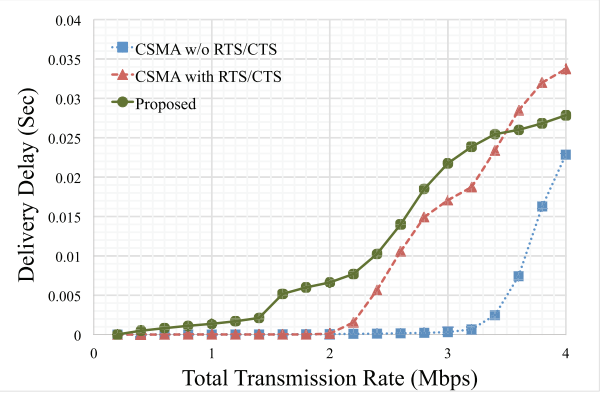


Fig. 14 Delivery delay (grid topology).

several inconsistent states occurred, e.g., more than two *active* nodes existed simultaneously among two-hop neighbors. We also note that, from the same reason, the delivery delay raises even with low transmission rate, as shown in **Fig. 14**.

5. Discussion on Fairness Issue

In designing MAC protocols, fairness is one of the most important issues to be considered. In the case of CSMA, fairness is assured in the sense that every node in a collision range obtains its transmission right with the equal probability. However, the current design of the proposed method does not consider fairness in any sense. This is essentially because, as we mentioned in Section 2, the proposed method is designed as a distributed implementation of 'Max-weight' scheduling. Note that Max-weight scheduling is valuable and well-recognized in that it is the optimal scheduling proved in theory. However, Max-weight scheduling itself does not consider fairness. Consequently, to be a practical MAC protocol, we have to devise a new way to treat fairness that works in the framework of Max-weight scheduling. This is one of the common future challenges between Max-weight scheduling and the proposed method. For example, if a node has one packet in its queue while its two-hop neighbors have more packets, and the state lasts for a long time, then the node will not able to transmit the packet even after that. This problem is serious in practice, and its solution is an essential challenge in the framework of Max-weight scheduling.

6. Conclusion

In this paper, we propose a new queue-length based distributed scheduling scheme that accelerates CSMA-based Wireless Mesh Networks. Each node propagates its queue-length information to every node in its two-hop neighborhood, and the node that has the largest queue-length in its two-hop neighborhood is allowed to transmit data frames. Through evaluation, we show that the proposed scheme significantly outperform the existing MAC technique used in CSMA-based WMNs.

One of the problems in this scheme is that the delivery ratio does not reach 100% in low-traffic cases. To develop a method to improve the problem would be an important future work topic that would have practical benefits.

References

[1] Akyildiz, I.F. and Wang, X.: *Wireless Mesh Networks*, John Wiley & Sons, Ltd, Publication (2009).
[2] Karn, P.: MACA – A New Channel Access Method for Packet Radio, *ARRL/CRRL Amateur Radio 9th Comp. Net. Conf.*, pp.134–140 (1990).
[3] Bharghavan, V. et al.: MACAW: A Media Access Protocol for Wireless LANs, *Proc. ACM SIGCOMM 1994* (1994).
[4] Sobrinho, J.L., de Haan, R. and Brázio, J.M.: Why RTS-CTS Is Not Your Ideal Wireless LAN Multiple Access Protocol, *Proc. WCNS'05* (2005).
[5] Xu, K., Gerla, M. and Bae, S.: Effectiveness of RTS/CTS Handshake in IEEE 802.11 Based Ad Hoc Networks, *Ad Hoc Networks*, Vol.1 No.1, pp.107–123 (2003).
[6] Tariquea, M., Tepeb, K.E., Adibic, S. and Erfanib, S.: Survey of multipath routing protocols for mobile ad hoc networks, *Journal of Network and Computer Applications*, Vol.32, No.6, pp.1125–1143 (2009).
[7] De. Couto, D., Aguayo, D., Bicket, J. and Morris, R.: A High-Throughput Path Metric for Multi-Hop Wireless Sensor Networks, *MOBICOM* (2003).
[8] Draves, R., Padhye, J. and Zill, B.: Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks, *Proc. 10th Annual International Conference on Mobile Computing and Networking* (*MOBICOM2004*), pp.114–128 (2004).
[9] Marina, M.K., Das, S.R. and Subramanian, A.P.: A topology control approach for utilizing multiple channels in multi-radio wireless mesh networks, *Computer Networks*, Vol.54, pp.241–256 (2010).
[10] Raniwala, A. and Chiueh, T.: Architecture and Algorithms for an IEEE 802.11-based Multi-channel Wireless Mesh Network, *Proc. IEEE INFOCOM'05*, pp.2223–2234 (2005).
[11] Kanaoka, H. and Yoshihiro, T.: Combining Local Channel Selection with Routing Metrics in Multi-channel Wireless Mesh Networks, *IPSJ Journal of Information Processing* (*JIP*), Vol.23, No.2 (2015).
[12] Sheshadri, R.K. and Koutsonikolas, D.: Comparison of Routing Metrics in 802.11n Wireless Mesh Networks, *The 32nd IEEE International Conference on Computer Communications* (*INFOCOM'13*) (2013).
[13] Tassiulas, L. and Ephremides, A.: Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks, *IEEE Trans. Automatic Control*, Vol.37, No.12, pp.1936–1948 (1992).
[14] Lin, X., Shroff, N.B. and Srikant, R.: A Tutorial on Cross-Layer Op-

timization in Wireless Networks, *IEEE Journal on Selected Areas in Communications*, Vol.24, No.8 (2006).
[15] Nardelli, B., Lee, J., Lee, K., Yi, Y., Chong, S., Knightly, E.W. and Chiang, M.: Experimental Evaluation of Optimal CSMA, *Proc. of INFOCOM'11* (2011).
[16] Bao, L. and Garcia-Luna-Aceves, J.J.: A New Approach to Channel Access Scheduling for Ad Hoc Networks, *Proc. 7th Annual International Conference on Mobile Computing and Networking* (*Mobicom'01*), pp.210–221 (2001).
[17] Rhee, I., Warrier, A., Aia, M. and Min, J.: Z-MAC: A Hybrid MAC for Wireless Sensor Networks, *Proc. SenSys'05* (2005).
[18] Rajendran, V., Obraczka, K. and Garcia-Luna-Aceves, J.J.: Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks, *Proc. Sensys'03* (2003).
[19] Rhee, I., Warrier, A., Min, J. and Xu, L.: DRAND: distributed randomized TDMA scheduling for wireless ad-hoc networks, *Proc. 7th ACM International Symposium on Mobile ad hoc Networking and Computing* (*MobiHoc'06*), pp.190–201 (2006).
[20] Wu, Z. and Raychaudhuri, D.: D-LSMA: Distributed Link Scheduling Multiple Access Protocol for QoS in Ad-hoc Networks, *Proc. IEEE Global Telecommunication Conference* (*Globecom2004*) (2004).
[21] Space Time Engineering, Network Simulator Scenargie, available from ⟨https://www.spacetime-eng.com/en/⟩ (accessed 2015-02).

Toshiki Takeda received his B.E. and M.E. degrees from Wakayama University in 2013 and 2015, respectively. He is currently working with Mitsubishi Electric Information Network Corporation.

Takuya Yoshihiro received his B.E., M.I. and Ph.D. degrees from Kyoto University in 1998, 2000 and 2003, respectively. He was an assistant professor at Wakayama University from 2003 to 2009. He has been an associate professor in Wakayama University from 2009. He is currently interested in graph theory, distributed algorithms, computer networks, wireless networks, medical applications, bioinformatics, etc. He is a member of IEEE, IEICE, and IPSJ.