Forwarding Mechanism Using Prioritized Forwarders for Opportunistic Routing

Taku Yamazaki[†], Ryo Yamamoto[††,‡], Takumi Miyoshi[‡‡,‡], Takuya Asaka[*], and Yoshiaki Tanaka[†,‡]
[†] Department of Communications and Computer Engineering, Waseda University
3-4-1 Okubo, Shinjuku-ku, Tokyo, 169–8555 Japan
[††] Graduate School of Informatics and Engineering, The University of Electro-Communications
1-5-1 Chofugaoka, Chofu-shi, Tokyo, 182–8585 Japan
[‡] Global Information and Telecommunication Institute, Waseda University
3-4-1 Okubo, Shinjuku-ku, Tokyo, 169–8555 Japan
[‡‡] College of Systems Engineering and Science, Shibaura Institute of Technology
307 Fukasaku, Minuma-ku, Saitama-shi, Saitama, 337–8570 Japan
[*] Faculty of System Design, Tokyo Metropolitan University
6-6 Asahigaoka, Hino-shi, Tokyo, 191–0065 Japan
Email: taku_yamazaki@aoni.waseda.jp, ryo_yamamoto@is.uec.ac.jp,
miyoshi@shibaura-it.ac.jp, asaka@tmu.ac.jp, ytanaka@waseda.jp

***Abstract*—In ad hoc networks, backoff-based OR (opportunistic routing) protocols, which autonomously selects a forwarder among the potential forwarders based on a random backoff time, have been proposed. However, each potential forwarder must wait for the backoff time to avoid packet collisions among receivers. In addition, the terminal density strongly affects the performance since the backoff-based OR protocols improve the performance by using multiple terminals. In this paper, we propose a novel OR protocol called PRIOR (prioritized forwarding for opportunistic routing). In PRIOR, a terminal called prioritized forwarder, which forwards packets without using a backoff time, is selected among neighbours. In addition, we propose a hop-by-hop retransmission control that performs the retransmission terminal selection on the basis of a neighbour relation with the PFs. Moreover, we introduce an explicit acknowledgement mechanism on the basis of the difference of hop counts to alleviate the bad effect of the retransmission control in dense environments. Finally, we evaluate PRIOR in comparison with conventional backoff-based OR protocols in computer simulation.**

***Index Terms*—ad hoc networks, opportunistic routing, prioritized forwarder, hop-by-hop retransmission control.**

I. Introduction

Ad hoc networks are formed distributed by mobile terminals such as smartphones without relying on any infrastructure. However, the topologies of the network vary over time due to unstable wireless communication and terminal mobility.

To adapt to the variations, broadcast-based forwarding protocols called OR (opportunistic routing) [1] that exploit the broadcast nature of wireless communications have been proposed. They can forward packets by using multiple receivers without relying a specific route.

Backoff-based OR protocols [2]–[4] that make a forwarding decision based on a backoff time calculated by hop counts have been proposed for ad hoc networks and wireless sensor networks. However, they may increase an unnecessary packet forwarding since their potential forwarder cannot always forward a packet even if the potential forwarder is the closest to the destination among the receivers. Moreover, the terminal density strongly affects their performance since they require enough neighbours to forward packets.

In this paper, we propose a novel OR protocol named PRIOR (prioritized forwarding for opportunistic routing) that selects a PF (prioritized forwarder) on hop-by-hop. The PF, which is selected by each forwarder among the neighbours, forwards packets without using a backoff time. In addition, we propose a sigmoid-based backoff time calculation that the terminal closest to the destination is always prioritized to forward packets among the receivers. Moreover, we propose a hop-by-hop retransmission control that performs the retransmission terminal selection on the basis of a neighbour relation with PFs to improve the performance in sparse environments. Furthermore, we introduce explicit ACK (acknowledgement) based on difference in hop counts to alleviate the bad effect of retransmission control in dense environments.

II. Related Work

In this section, we take notice of backoff-based OR protocols [2]–[4] to discuss an OR for mobile ad hoc networks. Note that, we will consolidate and/or alter names from the original papers to make the comparison easily among them.

The backoff-based OR protocols have two routing phases: discovering a destination called “discovery phase”, and forwarding reply packets and data packets called “data phase” respectively. For the routing, every terminal has a table called “cost table”, which has entries that contain at least an address, hop count, sequence number, and lifetime. Each receiver records or updates the entry of the cost table by using packets only from the reverse path. Therefore, the protocols require to use a bidirectional flow to do the procedure. In the discovery phase, if a source does not have a cost entry of a destination, it performs a request packet flooding towards the destination.

When the destination receives the request packet, it broadcasts a reply packet towards the source. Here, each receiver forwards the reply packet with the same way as a data packet since at least one reverse path has been already established. In the data phase, upon receiving a reply packet or a data packet, each receiver r calculates a difference in hop counts δ_r,

$$\delta_r = h_{rd} - (h_{id} - 1) \quad (1)$$

where h_{id} denotes the hop count between the previous forwarder i and the destination d that was recorded as a hop count to the destination on the packet during the previous packet forwarding. h_{rd} denotes a hop count between the receiver r and the destination d recorded in receiver r's cost table. Ideally, if the packet regularly traversed one by one, the hop count h_{rd} will be equal to the hop count h_{id} minus 1. Therefore, the receiver r can estimate how many hops close or far to the destination by calculating the difference δ_r. After that, if the potential forwarder receives the same packet during the backoff time, the potential forwarder regards the packet as an implicit ACK. Hence, it cancels the packet forwarding because the other terminal already forwarded the packet. The specifications of these protocols are described in the following.

SSR (self-selective routing) [3] that calculates a backoff time on the basis of two types of increasing function has been proposed. SSR calculates a fixed backoff time according to whether $\delta_r \leq 0$ or not in order to impose the fixed delay on detour terminals. In addition, it adds the random backoff time based on increasing functions. In SSR, a forwarder broadcasts an explicit ACK to cancel unnecessary packet forwarding when it receives a forwarded packet. The destination also broadcast the explicit ACK packet instead of the data packet.

SRP (self-selecting reliable routing protocol) [3], the enhanced variant of SSR has been proposed. SRP gives receiver on regular path ($\delta_r = 0$) the highest priority in order to avoid the variation of the hop count as much as possible. In addition, a forwarder of the previous packet calculates the smallest random backoff time in order to avoid collisions among the previous forwarders. Moreover, in data phase, SRP uses TTL that is set to h_{sd} plus $\log_2 h_{sd}$ to reduce unnecessary detours. Unlike SSR, when a receiver r receives a packet with δ_r ranging $\delta_r > 1$, it ignores the packet since SRP has to make a route repairing. In other words, SRP does not use more than 2-hop detours without performing the route repair. In SRP, forwarder twice retransmits a forwarded packet if it has not acknowledged the packet. When the attempt fails, it performs a route repair which increases the hop count on its table and the one recorded in the packet by 2.

LFBL (Listen first, broadcast later) [4] that can choose a backoff time calculation from several methods. In the paper, DVR (distance + variance + random) method is recommended though its specification is not described in the original paper. Therefore, we will describe LFBL based on a slotted random method. The slotted random method separates potential forwarders into two groups according to whether detour or not. If not, the potential forwarder belongs to the first slot and it adds the backoff time which is randomized in the slot. If the potential forwarder is detour terminal, it belongs to the second slot, and it adds the fixed plus random backoff time in order to make the priority low. In addition, LFBL restricts the implicit ACK condition that potential forwarders regard a data packet as an implicit ACK when it receives the same data packet with δ_r ranging $\delta_r > 0$. When δ_r is same as or less than 0, it re-calculates the backoff time b_r. Unlike SSR and SRP, LFBL do not have a function of explicit ACK. Therefore, potential forwarders cancel the packet by only using implicit ACK.

In the conventional protocols, a potential forwarder cannot always forward a packet even if it is the closest to the destination among the others. Therefore, they may cause the unnecessary detours. In addition, although SSR and SRP use an explicit ACK to reduce unnecessary packets, their receiver cannot know whether the sender is further to the destination than the receiver or not. Namely, the packet may not be forwarded correctly and it may excessively cancel packet forwarding. SRP eliminates over 2-hop detoured terminals instead of having a route repair mechanism. The elimination can avoid the use of excessive detoured path and may stop the necessary packet forwarding by restricting a forwarding area. Furthermore, the conventional protocols have a disadvantage that the performance strongly depends on a terminal density since they are basically designed to forward packets using multiple forwarders.

III. Prioritized Forwarding for Opportunistic Routing

We propose a novel OR protocol called PRIOR (prioritized forwarding for opportunistic routing). In PRIOR, each forwarder specifies a single terminal as a PF (prioritized forwarder) among the neighbours that forwards a packet without using a backoff time. In addition, we propose a sigmoid-based backoff time calculation that always prioritizes the closest terminal to a destination among receivers.

A. Forwarding Mechanism Using Prioritized Forwarders

As mentioned above, PRIOR uses a PF that is able to forward packets without using a backoff time. The forwarder explicitly selects a PF from among its neighbours when it forwards a packet. PRIOR also requires several conditions similar to the conventional OR protocols. PRIOR also requires a broadcast medium and to be used under a bidirectional flow since each terminal updates a cost entry using a packet on the reverse path. In addition, PRIOR requires a sender address to record or update PFs unlike the conventional OR protocol. In general, datalink layer protocols add the sender address to a frame header before the frame is transmitted. Therefore, PRIOR obtains the sender address from the datalink layer protocol if available. Otherwise, it adds the sender address into a packet header.

In PRIOR, every terminal has a cost table that contains cost entries including a destination address, hop count, PF address, sequence number, and lifetime. The update sequence of the entry is similar to the conventional protocols. First, each terminal checks whether it has an entry or not when it receives

a packet. If it does not have the entry, it records the entry to the source by using the information on the packet. If it already recorded the entry before, it checks the novelty of the entry by checking the sequence numbers. If the sequence number of the entry is smaller than the one of the received packet, it updates the entry since the packet is new one. When these sequence number are the same, it checks the hop count. If the hop count to the source on its table is larger than the one of the packet traversed, the terminal also updates the entry since the packet traversed the shorter path.

PRIOR also has a discovery phase and data phase. In the discovery phase, if a source does not have a cost entry to a destination, the source performs a request packet flooding towards the destination and waits the reply packet from the destination. The request packet contains a source address, destination address, sender address, traversed hop count, sequence number, and TTL. On receiving the request packet, the receiver records or updates its cost entry. When the destination receives the request packet, it broadcasts a reply packet towards the source only once. Here, the reply packet is forwarded with the same way of the data packet since the reverse forwarding path has already been established at least during the request packet flooding. If the source does not receive any reply packet in a certain time, it performs the request packet flooding with increased sequence number and TTL to expand the flooding area.

In the data phase, each terminal autonomously makes a forwarding decision upon receiving a reply or data packet. These packets contain a source address, destination address, sender address (if it is necessary), PF address, hop count to the destination, traversed hop count, sequence number.

Figure 1 shows an example of the packet forwarding procedure in PRIOR. Upon receiving a data or reply packet, each receiver checks whether the packet has already been received or not. If it has not, it checks two conditions. First, if it coincides with a PF in the packet, it becomes a PF and it forwards the packet without using a backoff time. If it is not the PF, it becomes a potential forwarder and it waits the backoff time based on δ_r. Note that, We describe the detail of the backoff time calculation in Section **III-B**. After the backoff time calculation, the potential forwarder receives the same packet during the backoff time, it checks δ_r of the packet according to Eq. (1). If δ_r is larger than 0, the potential forwarder cancels own packet forwarding since it regards the packet as an implicit ACK. Otherwise, it ignores the packet.

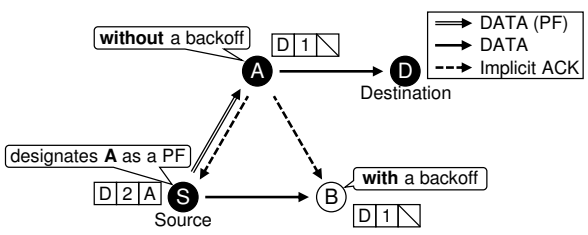

Fig. 1. Example of forwarding sequence in PRIOR.

B. Sigmoid-based Backoff Time Calculation

The conventional OR protocols may cause an extension of hop counts and an increase of unnecessary packet forwarding since their potential forwarder closest to the destination cannot always forward first. Then, we propose a novel sigmoid-based backoff time calculation mechanism that autonomously prioritize the terminal closest to the destination among potential forwarders.

In general, we assume that data packets are forwarded bidirectionally between a source and destination. Then, we expect that every forwarder can update their cost entries accurately and regularly. If forwarders directly and stably communicate each other, they have a neighbour relation. Then, receiver r calculates δ_r approximately ranges $0 \leq \delta_r \leq 2$. In contrast, if a packet is forwarded via a shortcut or an over 3-hop detoured path due to temporary factors such as the better radio environments and so on, a receiver r calculates δ_r approximately ranges $\delta_r < 0$ or $\delta_r > 2$. Therefore, a joint packet reception probability with the same δ_r ranges in the range of $0 \leq \delta_r \leq 2$ will also be much higher than the one in the range of $\delta_r < 0$ or $\delta_r > 2$. Hence, if the receiver r calculates δ_r in the range of $0 \leq \delta_r \leq 2$, it requires a large window for the random backoff time to reduce collisions. In contrast, if the receiver r calculates δ_r in the range of $\delta_r < 0$ or $\delta_r > 2$, it requires smaller window for the random backoff time to reduce collisions.

By exploiting the above characteristics, receiver r calculates the fixed backoff time based on a sigmoid function that converges to 0 or 1 by decreasing or increasing δ_r respectively, and adds a random backoff time based on the vertical change on the sigmoid function. The fixed backoff time $\varsigma_r(\delta_r)$ is calculated as

$$\varsigma_r(\delta_r) = \frac{1}{1 + \exp\Big(-(\delta_r - 0.5)\Big)} \tag{2}$$

The terminal r calculates the fixed backoff time $\varsigma_r(\delta_r)$ that decides the priority based on the sigmoid function according to δ_r. Namely, the fixed backoff time $\varsigma_r(\delta_r)$ is lower bound of the backoff time for each δ_r. For maximizing the random backoff time of terminals whose δ_r is 0, we subtract 0.5 from δ_r. Moreover, terminal r adds a random backoff time to the fixed backoff time according to a vertical change in the function. Then, the random backoff time $\mu_r(\delta_r)$ is calculated as

$$\mu_r(\delta_r) = u\Big(\varsigma_r(\delta_r + 1) - \varsigma_r(\delta_r)\Big) \tag{3}$$

where u denotes uniform random number of $(0, 1)$. The random backoff time $\mu_r(\delta_r)$ is based on the difference between $\varsigma_r(\delta_r + 1)$ and $\varsigma_r(\delta_r)$. Therefore, $\varsigma_r(\delta_r)$ plus $\mu_r(\delta_r)$ will always be smaller than $\varsigma_r(\delta_r + 1)$. By using $\varsigma_r(\delta_r)$ and $\mu_r(\delta_r)$, the backoff time b_r is calculated as

$$b_r = T_{\max}\Big(\varsigma_r(\delta_r) + \mu_r(\delta_r)\Big) \tag{4}$$

where $T_{\max}$ denotes the maximum backoff time. As mentioned above, the sigmoid function converges to 0 or 1. Therefore, the

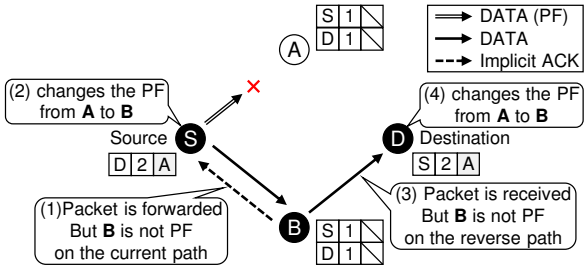

Fig. 2. Update procedures of a prioritized forwarder.

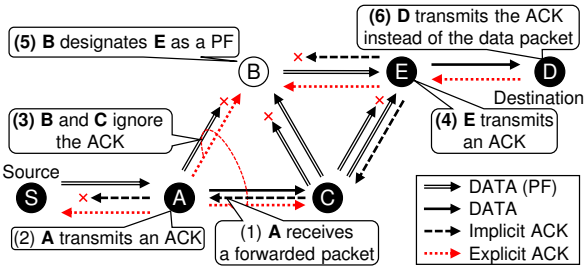

Fig. 3. Examples of explicit acknowledgement.

receiver r calculates the backoff time b_r with the product of the scaling factor $T_{\max}$ and the summation of the fixed backoff time $\varsigma_r(\delta_r)$ and the random backoff time $\mu_r(\delta_r)$.

C. Updating Prioritized Forwarders

In PRIOR, terminals are required to check the existence of PF in its neighbour range to use the PF correctly. However, the PF becomes obsolete since it might move out to an ineligible position to forward packets due to the topological changes in mobile environments. Therefore, terminals are required to adapt the topological changes by updating PFs appropriately. To realize it, every terminal observes that the transmitted packet is forwarded by the PF on the current path or not and they checks that the received packet come from the PF on the reverse path or not. Figure 2 shows an examples of the update procedure of PF for the current and reverse path.

First, when a forwarder receives the same packet after the packet forwarding, it checks whether the packet is forwarded by PF or not. If the packet is forwarded by PF, it does nothing. Otherwise, it alters the PF to the destination by the sender of the received packet.

When the terminal receives the packet, it checks the sender of the packet. If the packet is forwarded by the PF of the reverse path, in other words, the PF to the source, the terminal does nothing. Otherwise, the terminal alters the PF to the source by the sender of the received packet.

IV. Extensions for PRIOR

In this section, we propose several extensions for PRIOR to alleviate the performance degradation in a particular situation: (1) area restricted hop-by-hop retransmission control and (2) explicit ACK mechanism on the basis of the difference of hop counts. The hop-by-hop retransmission control restricts the retransmission area on the basis of a neighbour relation with the PF to improve the performance in sparse environments. The explicit ACK mechanism cancels unnecessary packet forwarding and retransmissions on the basis of the difference of hop counts between the sender and each receiver to decrease the network load.

A. Restricted Hop-by-Hop Retransmission Control

To overcome the performance degradation in sparse environments, we introduce an extension of hop-by-hop retransmission control into PRIOR. The retransmission control restricts the retransmission area on the basis of the neighbour relation with the PF to suppress retransmitting packets excessively.

First, a forwarder checks the neighbour relation with the PF after a packet forwarding. Then, if the forwarder is the PF or a neighbour of the PF and its δ_r is same as or smaller than 1, it initiates a retransmission control. Note that, the terminal checks the hop count to the PF to examine a neighbour relation with the PF. If the hop count to the PF on its table is 1, the terminal regards the PF as a neighbour. Otherwise, it is not a neighbour of the PF, and hence it does not initiate the retransmission and discards the kept packet. The terminal calculates the retransmission backoff time since it should wait a regular packet forwarding and avoids the collisions. Therefore, the retransmission backoff time is set to about 3 times as long as the maximum backoff time plus the backoff time b_r. When the forwarder r receives the same packet until the backoff time, it calculates δ_r. If δ_r is same as or less than 0, it regards the packet as an implicit ACK and finishes the retransmission control. If the forwarder has not received the same packet during the backoff time, it retransmits the packet since it assumes that there is no forwarder. After that, the forwarder increases a retransmission count and re-calculates the backoff time. Then, the retransmission backoff time is set to large enough to avoid the current packet forwarding. If the retransmission count reaches the threshold that represents the maximum retransmission count, the terminal finishes the retransmission control, discards the kept packet, and ignores the packet since then.

B. Hop Count Based Explicit Acknowledgement

In the backoff-based OR protocols, a forwarder basically acknowledges a received packet by receiving a duplicated packet. However, it consumes network resources since forwarders forward or retransmit unnecessary packets when it fails to receive the duplicated packet. To solve the problem, SSR and SRP perform an explicit ACK transmission. However, the explicit ACK has disadvantages that all receivers indiscriminately cancel packet forwarding even if they have not forwarded the packet correctly. Therefore, it may cause a deadlock problem. Moreover, PRIOR increases the number of transmitted packet and the network load if it enables the retransmission control in dense environments since the forwarder may fail to acknowledge the forwarded packet.

To overcome the disadvantages of conventional OR protocols' explicit ACKs and to alleviate the bad effect of retransmission control, we introduce an extension of explicit ACK mechanism on the basis of the difference of hop counts to suppress unnecessary packet forwarding and retransmissions. In PRIOR, ACK packet consists of a source address, destination address, hop count to the destination, and sequence number. Figure 3 shows examples of explicit ACK packet transmission for the backward area terminals and that for an isolated terminal.

If a terminal receives the same packet after the packet forwarding, it transmits an explicit ACK packet and the receiver r of the explicit ACK calculates δ_r. If δ_r is 0 and larger, it cancels the packet forwarding and ignores the packet since then. If δ_r is less than 0, the terminal ignores the ACK packet to avoid the deadlock problem. When the destination receives the data packet, the destination always broadcasts the explicit ACK packet instead of the data packet even if the destination has already received the packet before.

If a potential forwarder has not forwarded packet yet and its neighbours have already finished packet forwarding, the regular forwarding by the potential forwarder will be unnecessary. Moreover, if the retransmission control is enabled, the potential forwarder happens to repeat the packet retransmission since it misunderstands that the packet is lost due to the absent of forwarding by the neighbours. Therefore, in this situation, there is a necessity that someone stops the unnecessary transmission. Hence, a PF transmits an explicit ACK if it receives the already acknowledged packet. By receiving the explicit ACK, the potential forwarder can acknowledge the packet even if all of neighbours have already forwarded the packet.

V. Performance Evaluation

A. Simulation Setup

In this paper, we evaluated the performance of SSR, SRP, LFBL, and PRIOR using QualNet [5] as a network simulator. Note that in what follows, we call PRIOR with the explicit ACK PRIOR-E. In this simulation, we observed the impact of varying the terminal density on the performance. Terminals were placed randomly in a 1,000 m × 1,000 m simulation area and the number of terminals was varied from 20 to 200 in steps of 20. Every terminal used IEEE 802.11b and disabled RTS/CTS (request to send / clear to send). Their transmission rates were set to 11 Mbps and the communication range was set to approximately 150 m. A random waypoint mobility model was used and the moving speed was randomly chosen from 0 m/s to 10 m/s without using a waiting time. We generated bidirectional traffic using UDP (user datagram protocol) and the pair of terminals was randomly chosen from all of them. The pair transmitted 1 Mbyte data, consisting of one thousand 1 kbyte packets, each other. We performed the simulation with and without the retransmission control. SSR and LFBL cannot use the retransmission control since they only use broadcast, which ARQ cannot be applied to in IEEE 802.11 MAC [6]. Furthermore, although SRP has a retransmission control function, it must be combined with a route repair. Therefore, as recommended in the paper [3], we set the maximum retransmission count n to 2 for SRP. Meanwhile, PRIOR and PRIOR-E can change the maximum retransmission count n, hence we set n to 0 and 3. Each cost entry on a cost table timeout was set to 3 seconds. The maximum backoff time was set to 5.0 ms in SRP, LFBL, and PRIOR. However, SSR does not have maximum backoff time since it calculates the backoff time based on linear increasing functions. Hence, in SSR, the upper bound of the backoff time was set to 2.5 ms when the δ_r is 0.

We evaluated these protocols in terms of their (a) packet transmission success rate, (b) average end-to-end delay, and (c) total number of transmitted packet. Note that, we do not include out of order packets in the simulation results. Moreover, the total number of transmitted packet is normalized to that of LFBL since the relative ratio is more appropriate than the actual value to showing the performance.

B. Simulation Results

Figure 4 shows that the packet transmission success rate of SSR and that of SRP are lower than the other protocols. This causes that they use an explicit ACK that indiscriminately cancels all packet forwarding within the ACK sender's range. Thus, they impair the path diversity that affects the packet transmission success rate. On the other hand, LFBL and PRIOR (n = 0) achieve higher packet transmission success rate than SSR and SRP. This is because they can get more opportunities to forward packets since their potential forwarders acknowledge only the kept packet by receiving the same packet without relying an explicit ACK mechanism. PRIOR-E (n = 0) achieves little lower packet transmission success rate than LFBL and PRIOR. Therefore, excessive cancelations are also happened in PRIOR-E even though it alleviates the bad effect of the explicit ACK by using a hop count. PRIOR (n = 3) achieves the highest packet transmission success rate among the protocols in the sparse environment. In such environments, every terminal does not have adequate number of neighbours to gain a forwarding path diversity in their communication range. Nevertheless, it improves the packet transmission success rate by improving the link reliability using a retransmission control. However,

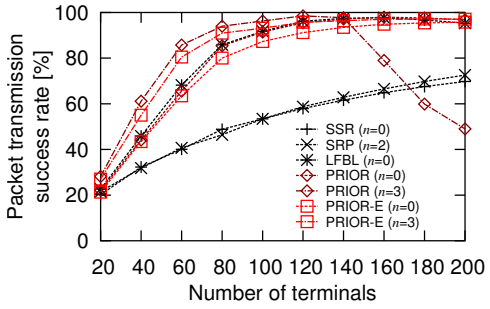

Fig. 4. Packet transmission success rate.

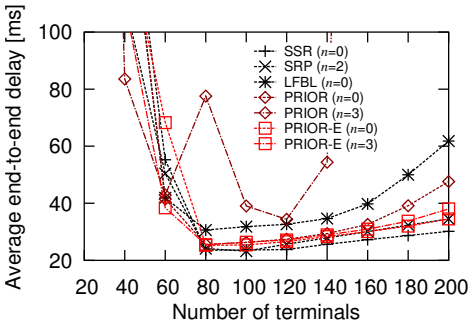

Fig. 5. Average end-to-end delay.

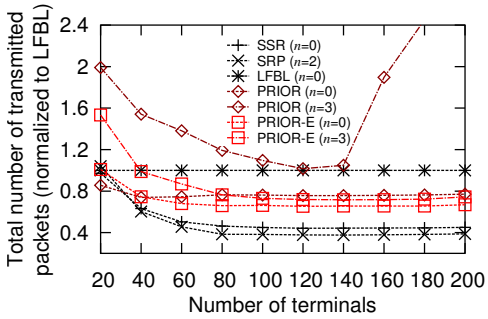

Fig. 6. Total number of transmitted packet.

PRIOR ($n = 3$) significantly decreases the packet transmission success rate in the dense environment. This is because that PRIOR requires that each terminal acknowledges packets only by an implicit ACK since PRIOR disables an explicit ACK mechanism. Therefore, each terminal requires to receive the same packet to cancel packet forwarding. Thus, if it fails to re-receive the packet, it causes misunderstandings and forwards unnecessary packet. As a result, PRIOR consumes more network resources in the dense environment and decreases the packet transmission success rate. In contrast, PRIOR-E ($n = 3$) does not happen to degrade the performance in dense environments. This is because that the network load decreases to cancel unnecessary packet forwarding by using the explicit ACK comparing with PRIOR ($n = 3$).

Figure 5 shows that SSR, PRIOR ($n = 0$), and PRIOR-E ($n = 0, 3$) achieve lower transmission delay than LFBL and PRIOR ($n = 3$). The result shows that SSR achieve a slightly lower transmission delay than the other protocols. As mentioned above, its explicit ACK indiscriminately cancels the packet forwarding and it does not have a function of the retransmission control. In addition, its detour terminals calculate a large amount of backoff time. As a result, it decreases the transmission delay since the detour forwarding, which includes necessary packet forwarding, is difficult to occur. On the other hand, PRIOR and PRIOR-E use PFs and the sigmoid-based backoff time calculation that always prioritizes the closest terminal to the destination among receivers. This can contribute to suppress the transmission delay increases without decreasing the transmission success rate since it becomes to select forwarders appropriately.

Figure 6 shows that SSR and SRP achieve the smallest total number of transmitted packets among the protocols. As mentioned above, they indiscriminately cancel the packet forwarding by explicit ACKs. Therefore, although they decrease the total number of transmitted packet, they degrade the packet transmission success rate. PRIOR-E ($n = 3$) has higher number of transmitted packet than SSR and SRP. However, it achieves significantly higher packet transmission success rate than SSR, SRP, and LFBL. Nevertheless, PRIOR-E achieves the lower number of transmitted packet than LFBL. This is because that PFs and the backoff time calculation improve the forwarder selection mechanism. Moreover, the increment ratio of the number of transmitted packet is lower than that of PRIOR even if it uses the retransmission control in the dense environment. This is because the explicit ACK of PRIOR-E accurately cancels the unnecessary packet forwarding.

VI. Conclusion

In this paper, we proposed a novel OR protocol called PRIOR that uses a PF for the better forwarder selection. In addition, PRIOR uses the sigmoid-based backoff time calculation that always prioritizes the potential forwarder closest to a destination among receivers. Moreover, we introduce the hop-by-hop retransmission control to alleviate an effect of terminal density. Furthermore, we proposed an extension of the explicit ACK on the basis of the difference of hop counts between a forwarder and its neighbours.

We evaluated the proposed method under various terminal densities by using the computer simulation. From the simulation results, the proposed method realizes both the higher packet transmission success rate and lower number of transmitted packets. Moreover, the hop-by-hop retransmission control contributes to the improvement of the end-to-end packet transmission success rate in most situations and alleviates an effect of terminal density. Currently, we only evaluated the impact of the terminal density changes. However, we assume that PFs of the proposed method is affected of the topological changes. Hence, there can be a room to discuss about the other simulation environment.

References

[1] H. Liu, B. Zhang, H.T. Mouftah, X. Shen, and J. Ma, "Opportunistic routing for wireless ad hoc and sensor networks: present and future directions," IEEE Commun. Mag., vol.47, no.12 pp.103–109, Dec. 2009.

[2] G.G. Chen, J.W. Branch, and B.K. Szymanski, "Self-selective routing for wireless ad hoc networks," Proc. IEEE Int. Conf. Wireless and Mobile Comput. Netw. and Commun. (WiMob 2005), vol.3, pp.57–64, Montreal, Canada, Aug. 2005.

[3] E. Gelenbe, P. Liu, B.K. Szymanski, and, C. Morrell, "Cognitive and self-selective routing for sensor networks," Computational Management Science, vol.8, no.3, pp237–258, Aug. 2009.

[4] M. Meisel, V. Pappas, and L. Zhang, "Listen first, broadcast later: topology-agnostic forwarding under high dynamics," Proc. Annual Conf. of Int. Tech. Alliance (ACITA 2010), London, UK, Sept. 2010.

[5] QualNet, website. http://web.scalable-networks.com/content/qualnet/

[6] ANSI/IEEE Std. 802.11, "Wireless LAN medium access control (MAC) and physical layer (PHY) specification," March 2012.