Regular Paper

Building Detour Paths to Avoid Local Congestion in MANETs

Kiyotaka Kaji[1] Takuya Yoshihiro[2,a)]

Received: May 4, 2017, Accepted: November 7, 2017

Abstract: MANETs (Mobile Ad-hoc NETworks) have been studied for a long time to provide multi-hop wireless communications among temporarily gathered mobile devices. However, due to the nature of wireless communications, the communication performance of a flow is easily degraded by even a single local congestion on the path. A simple solution for this problem is to utilize detour paths that avoid the congested area. However, to provide detour paths that avoid the area whenever a packet faces congestion is in fact a challenging task because the detour paths must avoid relatively large region in which harmful radio waves interfere. In this paper, we propose a routing scheme that proactively computes detour paths to avoid the congested area ahead of packets. Specifically, when a packet meets congestion, we assume that the next-next-hop node along the shortest path to the destination is the center of the congestion, and forward the packet to its destination without visiting the node neighboring the congestion central node. We designed an algorithm that builds the secondary routing tables that works to provide the detour paths on top of the distributed link-state routing scheme.

Keywords: MANETs, detour paths, multi-path routing, fault tolerance

1. Introduction

Mobile Ad-hoc NETworks (MANETs) have been extensively studied as a temporal infrastructure to exchange mobile data in a multi-hop distance using wireless radio communications. Because many studies suppose that MANETs are used by general users for exchanging data, the well-populated Wi-Fi that works on the license-free bands is one of the major communication standard used as the PHY and MAC protocols. Over those, several traditional routing protocols have been developed so far. There are two major approaches: one is proactive routing such as OLSR [1], which always maintains routing tables for every destination. The other is reactive routing such as AODV [2], which computes routing table on demand by searching paths via Route Request (RREQ) and Route Reply (RREP) messages.

However, in the multi-hop networks that deploy the Wi-Fi standard, we significantly suffer from severe interference that degrades communication performance [3]. In such networks, we often observe local congestion hot spots that is invoked by the concentration of traffic and communication paths in a particularly local area. We also observe in real situations that several temporarily busy Wi-Fi APs causes to make a severe congestion hot spot. Note that, in multi-hop communications, even a single congested link in a path can be a bottleneck of the whole communication performance. To avoid this kind of performance bottleneck, multi-paths utilization is an essential approach.

From this reason, there are many proposals for multi-path routing in MANETs for both proactive and reactive schemes. For proactive scheme, Yi et al. proposed MP-OLSR (Multi-Path OLSR) in which each node computes multiple disjoint paths for each destination and the source node of a flow uses these multiple paths using source routing [5]. Marina, et al. proposed a multipath technique for the reactive routing scheme in which multiple RREP messages are sent back to the source node to build a disjoint paths set [6]. Many other variants of this kind of multipath routing have been proposed so far (See the survey paper for those [4]). Those schemes basically distribute traffic at the source node for load balancing so that they cannot use detour paths adaptively in face of detecting congestion.

There are several proposals that utilize multiple paths to backup the main path in face of link failure [7], [8]. However, another problem arises in this case that the alternative paths are often not able to avoid the area of congestion because they are usually computed as a node-disjoint paths set and are not designed to make detour to avoid an area that includes multiple nodes. To avoid congested area by rerouting packets after they meet the congestion, we require a new mechanism to provide larger detour paths that avoid local congested areas.

In terms of rerouting scheme, we may refer IPFRR (IP Fast Reroute) technique, which is a well-studied technique to strengthen fault-tolerance in wired networks. IPFRR recovers failure using precomputed alternative paths as fast as possible after the failure is detected so that users would not be aware of the failure. In many cases, an IPFRR scheme guarantees to recover a single link/node failure within a small amount of additional network configurations. For example, FIFR [9] prepares a

[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

secondary routing table to recover any single link/node failure, and NotVia [10] precomputes a set of tunneling paths to recover communications against any single link/node failure. However, in IPFRR, area failure (or interference) that is essential in wireless network is not considered because failure in wired networks occurs on each component such as a link, a node, or it's combinations. Although there is a study to migrate IPFRR into wireless networks [11], it also does not consider regional failure.

In this paper, we propose a new distributed algorithm that work on top of general link-state routing scheme to build a large detour paths that enable packets to avoid a certain congested area immediately after they detect congestion. By making a larger detour than the state-of-the-art IPFRR schemes (which avoid just a failed link or a node), the proposed scheme is applicable to wireless environment. Note that the congested area would be formed by a node that scatters harmful radio, or whose radio causes severe interference so called such as hidden-terminal effects, etc. We therefore suppose that the congested area has the 1-hop diameter centered by a causal node, and provide detour paths that are available to avoid such area whenever a packet meets congestion. Since our detour-path computation invoked at each node only depends on the information of two-hop neighbors, our scheme surely works on top of any link-state scheme including OLSR, and the computational complexity is sufficiently low.

The reminder of the paper is organized as follows: In Section 2, we present the proposed framework to forward packets using detour paths in face of congestion. In Section 3, we describe the algorithm to build the detour routing table that represents the alternative paths to avoid congested area. We also provide an complexity analysis of the algorithm. In Section 4, we evaluate the proposed scheme through simulations, and discuss the effectiveness of the proposed multi-path approach in Section 5. Finally, we conclude the work in Section 6.

2. The Proposed Scheme

2.1 Overview

We assume a multi-hop network where each node deploys a CSMA-based MAC protocol such as IEEE 802.11. Each node has a single Network Interface Card (NIC) that is configured on the common frequency channel. Each node maintains a (primary) routing table that enables packets to travel on the shortest paths to all destinations. Note that any type of routing protocols including such as proactive OLSR and reactive AODV can be used as the base routing protocol of our method as long as it computes the shortest paths among all node pairs to build its (primary) routing table.

We suppose that there is a locally congested area in which severe interference occurs, being desirable to be avoided when packets travel to their destinations. Note that we can meet such situation in practice. For example, sometimes a Wi-Fi access point is temporarily heavily loaded due to mobile users passing by. Because the current MANET routing protocols usually forward packets on the shortest-paths, end-to-end performance of the flows that go through such unfortunate local congestion would be significantly degraded. Since congestion is typically invoked by concentration of wireless radios generated by one or more busy nodes, we model the congestion area as a physically-formed circular region centered by the nearest busy node inferred in terms of hidden terminal effects, which is located two-hop away from the detecting node.

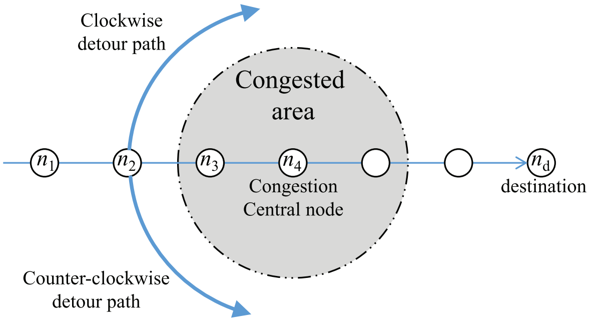


Fig. 1 The concept.

Our approach to avoid congested areas is to use local detour paths that bypass the congested area. We depict the idea in **Fig. 1**, where packets are to be forwarded from n_1 to the destination n_d along the shortest path. Suppose that node n_2 detects congestion on the next-hop link when it forwards a packet to n_3. In this case, n_2 infers that the place around the next-next-hop node n_4 on the shortest path would be the center of the congestion area, and forwards packets to avoid the area (Thus we call n_4 the *congestion central node* or just *central node*). Note that to avoid the next-next-hop node is reasonable because the next-next-hop node is a hidden terminal and it significantly degrades the performance if it transmits packets frequently. To avoid the congested area, n_2 forwards packets into either of two detour paths that lies along the boundary of the congested area with the direction of clockwise and counter-clockwise.

To have packets travel on a consistent path to its destination, we prepare an additional field in the packet header that holds the *central node*, and a node (such as n_2) writes the *central node* on the field when it detects congestion. The nodes on the detour path check the additional field to recognize whether each packet is the detouring packet or not. To forward the detoured packets into a consistent next-hop without creating loops, we prepare an additional routing table called *detour table*. From the destination and the central node in a packet header, nodes determine a consistent next-hop using the *detour table*. By using the clockwise or counter-clockwise detour path locally, packets travel to their destination without entering the congested area.

2.2 Table and Packet Structures

As mentioned in the previous section, we extend packet format as well as routing tables to enable packets to travel along detour paths consistently.

We extend the packet header by adding a field that holds an address of the central node. The field is initially empty (i.e., NULL) while packets travel along the shortest paths, and is filled with an address of central node when a node detects congestion and forwards the packets into detour paths. Naturally, the field is cleared when the packet finishes using the detour paths and returns to the shortest paths again.

We also extend the routing tables. Firstly, we add a field in the

Destination	Next-hop	Central Node

Fig. 2 Primary Routing Table Format.

Primary Next-hop	Central Node	1st Detour Next-hop	2nd One

Fig. 3 Detour Routing Table Format.

primary routing table so that it includes three fields as shown in **Fig. 2**: a destination node, a primary next-hop node, and a central node. Also, we add a *detour routing table* (or simply we call it a *detour table*) to query the next-hops for the detour paths. Specifically, the detour table includes the following four fields as shown in **Fig. 3**: a primary next-hop node, a central node, the 1st detour next-hop node, and the 2nd detour next-hop node.

When a packet to be forwarded into detour paths reaches to a node, the node first refers the primary table to obtain two values (i.e., the primary next-hop and the central node), and next with these two values we refer the detour table to obtain two detour next-hops. The detail of the forwarding process is described in Section 2.4.

2.3 Congestion Detection

When a node detects congestion on the primary next-hop link, the packets begin to use detour paths instead of using the shortest paths. To detect congestion on a link, we rely on the mechanism of MAC protocols. Specifically, we assume to run the proposed method on top of IEEE802.11 standard, and we use the retransmission count to detect congestion. We always compute the average of retransmission counts for the last k frames, and if it is larger than the predefined threshold value N_{ret}, we regard that the link is in the congested state. Note that frequent retransmissions not always mean congestion, but we regard it congestion because it is the state that requires alternative paths to forward packets anyway.

After the congestion detection, we regard that the congestion lasts for a certain time, i.e., packets that use the link as their primary next-hop will be all forwarded into the detour next-hops for the period of time predefined as $T_{duration}$.

2.4 Packet Forwarding

In a normal state without congestion, we use the primary routing table and packets travel along the shortest paths to their destination. In this section, we specify the behaviors when we use the detour paths. The behaviors of nodes that we explain are classified into three parts, i.e., A) the behavior when a packet begins to use a detour path, B) the behavior during a packet is traveling along the detour path, C) and the behavior when a packet finishes using the detour path.

A) As mentioned above, a packet begins to use a detour path when a node detects congestion in its primary next-hop link. For a packet that is traveling along the shortest path, nodes first refer the primary routing table. If the primary next-hop link is in the congested state, they retrieve the *central node* from the primary table, and stores it in the packet header. The nodes next refer the detour table using the primary next-hop and the central node, and find at most two detour next-hops. If two detour next-hops are found, the nodes can apply either of them as long as the corresponding link is not in the congested state. If only one detour next-hop is retrieved, the nodes apply it as long as the corresponding link is not in the congested state. If no detour next-hop is found, the nodes remove the central node from the packet header, and apply the primary next-hop.

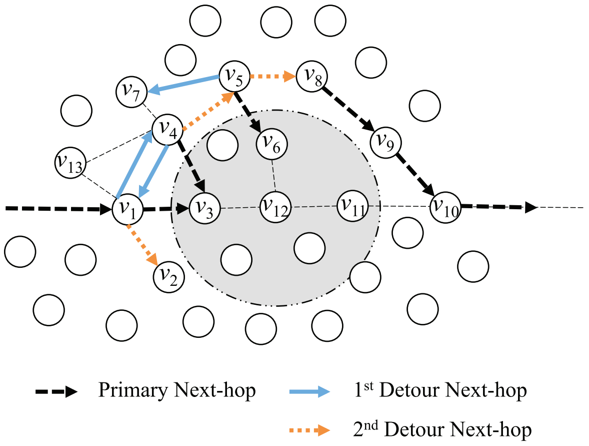


Fig. 4 Example of Detour Forwarding.

B) When a packet traveling along the detour path (i.e., a packet that has a central node in its packet header) reaches a node, the node checks whether the packet is to continue detouring or not. This is done by checking whether the primary next-hop for the packet is a neighbor of the central node written in the packet or not: if the condition meets, to avoid forwarding it into the congestion area, the node again refers the detour table and apply one of the detour next-hops. If the condition does not meet, we proceed the case C). In choosing the detour next-hop to apply, we have to be careful to forward packets toward the same direction (i.e., clockwise or counter-clockwise) as the previously forwarded direction of the packets. To do this, we use the information of previous hop: Specifically, we avoid using the next-hop that is a neighbor of the previous hop. Let n_p be the previous hop of a current node n_c, and n_1 and n_2 be two next-hop candidates retrieved from the detour table. If n_1 is the same node as n_p or one of n_p's neighbors, then n_1 is the same direction as n_p and would make a routing loop with high probability. Thus we have to choose n_2 in this case. Formally, we choose the next-hop in the following steps. First, retrieve the first next-hop candidate n_1 from detour table. Second, if it is neither n_p itself nor n_p's neighbor, we select n_1 as the next-hop of the packet. Otherwise, retrieve the second next-hop n_2 and select it as the next-hop. If no feasible detour next-hop exists, the node applies the primary next-hop.

C) When the primary next-hop is not a neighbor of the center node, nodes regard that the packet is no more in the congested area, and return the packets to the shortest paths. The concrete operation for this is that the node removes the central node from the packet header and forwards it using the primary routing table.

An example of the packet forwarding process is shown in **Fig. 4**, and the corresponding primary and detour tables of v_1, v_4 and v_5 are shown in **Tables 1–6**, respectively. In this example, suppose that v_1 is going to forward packets to v_{10}. When v_1 detects congestion in the next-hop link (v_1, v_3), by referring the primary table shown in Table 1 with destination v_{10}, v_1 regards the next-next-hop v_{12} as the congestion central node, and write it in

Table 1 Primary Table of v_1.

Destination	Primary Next-hop	Congestion Central Node
:	:	:
v_{10}	v_3	v_{12}
v_{11}	v_3	v_{12}
v_8	v_4	v_5
:	:	:

Table 2 Primary Table of v_4.

Destination	Primary Next-hop	Congestion Central Node
:	:	:
v_{10}	v_3	v_{12}
v_{11}	v_3	v_{12}
v_8	v_5	v_8
:	:	:

Table 3 Primary Table of v_5.

Destination	Primary Next-hop	Congestion Central Node
:	:	:
v_{10}	v_6	v_{12}
v_{11}	v_3	v_{12}
v_8	v_8	–
:	:	:

Table 4 Detour Table of v_1.

Primary Next-hop	Congestion Central	1st NH	2nd NH
:	:	:	:
v_3	v_{12}	v_4	v_2
v_4	v_5	v_3	v_{13}
:	:	:	:

Table 5 Detour Table of v_4.

Primary Next-hop	Congestion Central	1st NH	2nd NH
:	:	:	:
v_3	v_{12}	v_1	v_5
v_5	v_8	v_3	v_7
:	:	:	:

Table 6 Detour Table of v_5.

Primary Next-hop	Congestion Central	1st NH	2nd NH
:	:	:	:
v_6	v_{12}	v_7	v_8
v_4	v_3	v_7	v_8
:	:	:	:

the packet. Next, by querying the detour table with the primary next-hop v_3 and the central node v_{12}, v_1 finds that the 1st next-hop candidate is v_4, and forward the packet to v_4. On the next node v_4, by referring the primary table shown in Table 2, v_4 finds that v_3 is the primary next-hop for v_{10}. However, since v_4 knows that v_3 is a neighbor of the congestion central node v_{12}, which is written in the packet, v_4 determines to use detour paths. By referring the detour table shown in Table 5 with v_3 and v_{12}, v_4 find that the first next-hop candidate is v_1. However, since v_1 is the previous hop itself, v_4 use the second candidate v_5 to forward the packet. On the next node v_5, in the similar process, v_5 also refers its detour table. Since the first candidate v_7 is a neighbor of the previous hop, v_5 uses v_8 to forward packets. In v_8, the primary next-hop v_9 is no more the neighbor of the congestion central node v_{12}, so v_8 remove v_{12} from the packet and forward it to v_9. In this way, a packet travels to the destination v_{10} without entering the 1-hop area centered at v_{12}.

3. Detour-paths Computation

3.1 Algorithm to Compute Detour-table

We describe the algorithm that computes the detour table at each node. Note that the primary table is updated exactly same timing as the traditional link-state routing scheme, and the detour table is updated just after the primary table. In our algorithm, we assume that every node knows all of its neighbors, and the neighbors of them. Consequently, every node knows its two-hop neighbors, which are the nodes that are two-hop distant from the node. Note that this assumption is easily achieved by the traditional hello messages used in link-state routing protocols where each node periodically advertises its neighbors using hello messages. We denote the neighbor set of v by $\mathcal{N}_1(v)$, and the two-hop neighbor set of v by $\mathcal{N}_2(v)$. Also, let $\mathcal{N}_2^p(v)$ be the set of nodes in $\mathcal{N}_2(v)$ that are adjacent to $p \in \mathcal{N}_1(v)$.

In computing the detour table, each node v is required to find the farthest node from v along the boundary of the congested area in both clockwise and counter-clockwise directions. Our algorithm to do this for a single destination and a central node is shown in the following. To build the detour table, the following steps are executed for all possible pair of a destination n_d and a central node n_c.

Algorithm ComputeDetourTable

1. Compute $\mathcal{N}_2^p(v)$ where p is the primary next-hop of v for n_d.
2. Compute the candidate set of the detour nexthops $C(p)$, which is the set of nodes in $\mathcal{N}_1(v)$ that is adjacent to a node of $\mathcal{N}_2^p(v)$, and is not adjacent to n_c.
3. For each candidate $x \in C(p)$, we compute the *neighbor number* S_x, which is the number of x's neighbors in $C(p)$.
4. In $C(p)$, we choose a pair of nodes x and y as the 1st and 2nd detour next-hops where x and y are not adjacent with each other and the sum of the neighbor number $S_x + S_y$ is the smallest among all pairs in $C(p)$.
5. If no pair can be chosen in step (4), we choose the node x in $C(p)$ that has the smallest neighbor number S_x as the 1st detour next-hop. In this case, the 2nd detour next-hop is NULL.

Figure 5 illustrates the process to choose the detour next-hops, where n_1 would compute its detour next-hops, n_2 is the primary next-hop, and n_3 is the central node of the congestion area. The nodes in the right-side group $\mathcal{N}_2^{n_2}(n_1)$ represent the subset of two-hop neighbors of n_1 adjacent to n_2, and the nodes in the left-side group $C(n_1)$ represent the candidate set of detour next-hops on n_1. By finding the pair of nodes that has the least sum of neighbor numbers, we can choose two nodes that are the farthest from the $n_1 - n_2$ line in the both sides. By choosing the farthest nodes in the both sides, we intend to select the shortest paths that does not include the nodes located in the congested area. In the case of Fig. 5, the pair n_6 and n_7 is chosen, which is placed in the both end (upper and lower end) of the area in which the nodes in $C(n_1)$ exist. In this way, the proposed algorithm chooses two detour next-hops that carry packets along the boundary of congestion area in both directions.

3.2 Computational Complexity

In this section, we analyze the computational complexity of the algorithm ComputeDetourTable, which computes the detour table at each node. To analyze the complexity of the algorithm, we assume that each node maintains the list of neighbors, and if a neighbor is selected, the list of two-hop neighbors adjacent to it is directly obtained. Note that this is the same structure of information included in a hello message of link-state protocols.

In step 1 of ComputeDetourTable, we compute $\mathcal{N}_2^p(v)$ where p is the primary next-hop of v. From the assumed data structure in each node, this is done with $O(|\mathcal{N}_1(v)|)$ time. In step 2, we seek all nodes in $\mathcal{N}_2(v)$ and pick up the corresponding neighbors in $\mathcal{N}_1(v)$ to form $C(p)$. This process potentially includes to check all the neighbor and two-hop neighbor pairs, thus it takes $O(|\mathcal{N}_1(v)||\mathcal{N}_2(v)|)$ time. In step 3, we count the number of neighbors in $C(p)$ for every nodes in $C(p)$, which takes $O(|C(p)|^2)$. Since $C(p) \subseteq \mathcal{N}_1(v)$, this is $O(|\mathcal{N}_1(v)|^2)$. In step 4, we checks all node pairs, so we similarly obtains $O(|\mathcal{N}_1(v)|^2)$ time. From above, algorithm ComputeDetourTable takes $O(|\mathcal{N}_1(v)||\mathcal{N}_2(v)|)$ time in total. Since ComputeDetourTable runs for each primary next-hop, we require $O(|\mathcal{N}_1(v)|^2|\mathcal{N}_2(v)|)$ time to obtain the detour table.

Note that, the single-source shortest-path computation in sparse graph takes $O(V^2)$ time with Dijkstra's algorithm, where V is the number of nodes in the network. Since the number of two-hop nodes $|\mathcal{N}_2(v)|$ is usually far smaller than V, and the number of neighbors $|\mathcal{N}_1(v)|^2$ is approximated by $|\mathcal{N}_2(v)|$, the time complexity to compute detour table in our estimation is less than that of shortest-path computation.

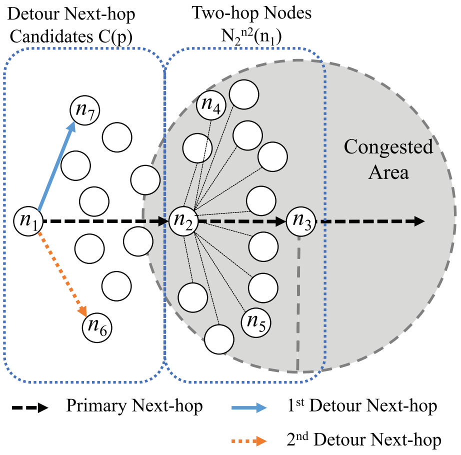


Fig. 5 Computing Detour Next-hops.

4. Evaluation

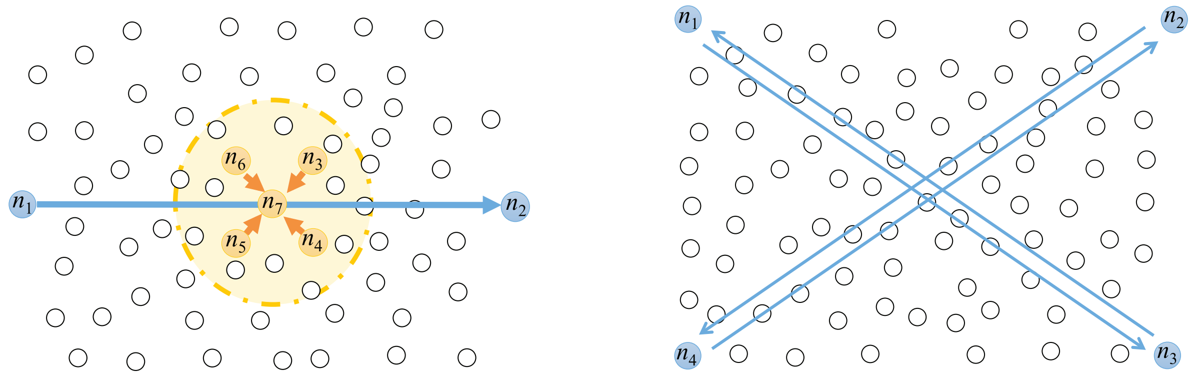
4.1 Methods

We evaluated the proposed method through simulation. We use the network simulator Scenargie [12] that implements up to date MAC and PHY models. The major objective of the evaluation is two folds. First of all, we show that the detour paths are surely available at each node whenever they detect congestion and wish to use them. This is the basic but important property of our scheme because the path formation is one of the main contribution of this work. The second one is to see the performance of our scheme in practical scenarios.

We designed two simulation scenarios considering practical scenes as follows. The first scenario is shown in **Fig. 6** (a), in which 150 nodes are placed randomly in a 1,500 × 1,500 [m] square field, and a busy access point (AP) is placed at the center of it. Specifically, we place n_7 that plays the role of an AP, and four nodes $n_3 - n_6$ transmits frames to n_7 continuously throughout the simulation time. In this environment, we generate a CBR (Constant Bit Rate) flow from the left end, i.e., n_1, to the right end, i.e., n_2. With several transmission ratio of the flow, we measured the throughput, delivery ratio and latency of the flow.

The second scenario is shown in Fig. 6 (b). In this scenario, we suppose the situation where several flows exist and their paths across with one another, where crossing point would be congested. Specifically, we place 150 nodes randomly in the same way as the first scenario, and generate four bi-directional CBR flows diagonally between $n_1 - n_3$ and $n_2 - n_4$. With the proposed scheme, we expect to reduce the congestion by the adaptive load balancing mechanism that utilizes detour paths.

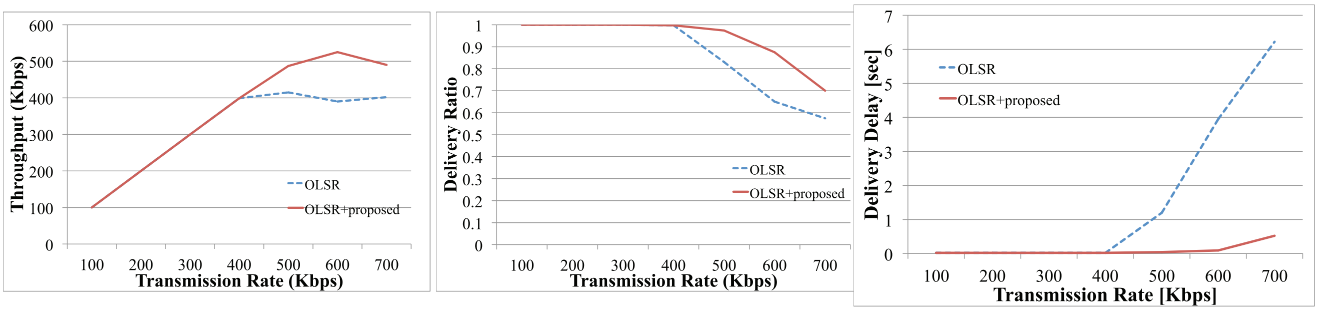
The parameters of the simulation are summarized in **Table 7**. We use IEEE 802.11g standard in 6 [Mbps] transmission speed. The transmission power is set to 10 [dBm] so that the radio is reachable to about 300 [m] distance. As a base routing protocol, we use OLSR implementation contributed by Niigata University, and we extend it to implement the proposed scheme. We use $N_{ret} = 3$ as the threshold of retransmission to judge congestion, and the congestion state lasts with $T_{duration} = 500$ [msec]. In the first scenario, we tried two transmission rates of the interfering



(a) Scenario 1

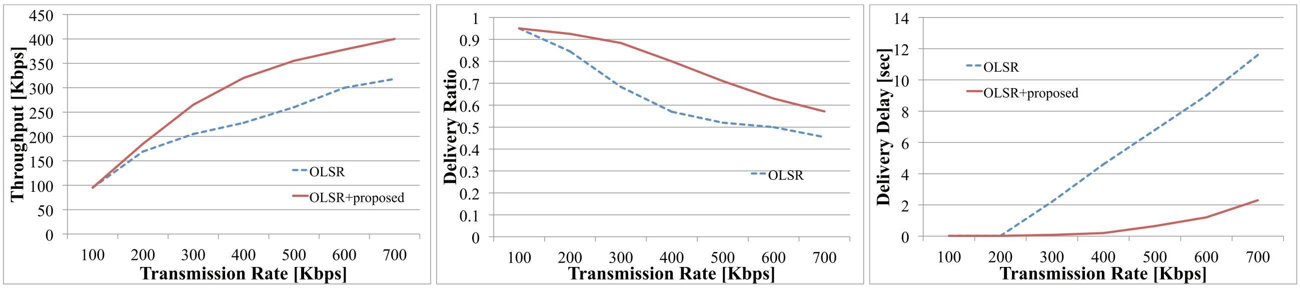
(b) Scenario 2

Fig. 6 Simulation Scenarios.



(a) Throughput (b) Delivery Ratio (c) Delivery Delay

Fig. 7 Results of Scenario 1 with 300 Kbps interference.



(a) Throughput (b) Delivery Ratio (c) Delivery Delay

Fig. 8 Results of Scenario 1 with 500 Kbps interference.

Table 7 Simulation Settings.

items	values
Field Size	1,500 × 1,500 [m]
# of nodes	150
Simulation Time	5 [min]
Routing Protocol	NU_OLSR
Hello Interval	2 [sec]
TC Interval	5 [sec]
MAC and PHY	IEEE 802.11g
Link Speed	6 [Mbps]
Transmission Power	10 [dBm]
Transport Protocol	UDP
Flow Type	CBR (Constant Bit Rate)
Packet Size	512 [Bytes]
Transmission rate of $n_3 - n_6$	300 and 500 [Kbps]
Congestion Detection Threshold N_{ret}	3
Congestion Lasting Time	500 [msec]

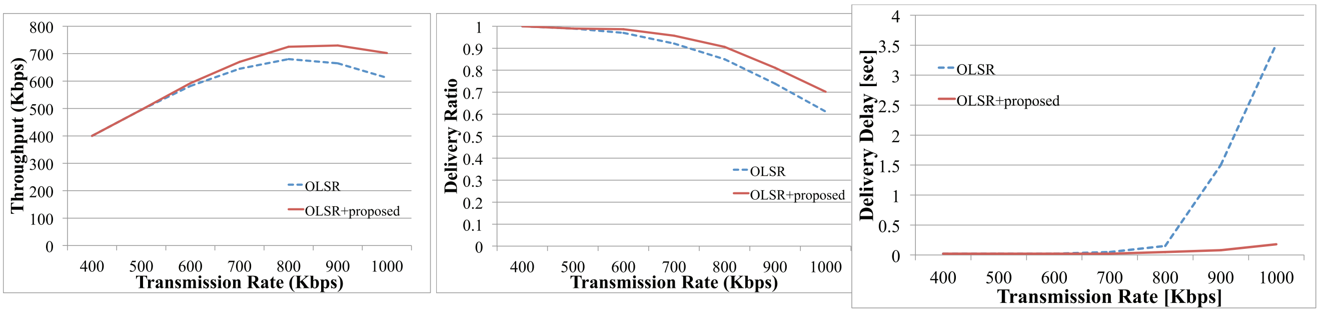
nodes $n_3 - n_6$, 300 [Kbps] and 500 [Kbps] at each node, to examine the low-load and high-load situation. Simulation time is 5 [min] where we first wait for 1 minute for the convergence of the routing protocol and measure the performance for the rest 4 minutes. We run 10 simulations for each flow rate and compare the average of them. Note that we use the different node placements for each trial.

4.2 Results

The results of the first scenario with 300 [Kbps] interference rate are shown in **Fig. 7**. From Fig. 7 (a)(b), we see that the throughput of the proposed method improves, and accordingly the delivery ratio also improves. Figure 7 (c) shows the delay performance where the conventional OLSR rapidly raises around 400 [Kbps] transmission rate, meaning that the network exceeds the capacity and is saturated in the conventional method. Note that the performance improvement is achieved by utilizing multiple paths at the entrance of the congested area. In the simulation, we observed that most of the packets were travelling along with two paths, i.e., the shortest paths and the detour paths invoked at the specific node placed at the enterance of the congested area. From this, we firstly conclude that the detour paths are surely build by the proposed method, and available at the node that requires them. We also conclude that the proposed method extends the capacity of the network by utilizing the detour paths when the traffic exceeds the capacity of the primary path. In this scenario, we saw that the load-balancing function works over two explicit paths, i.e., the shortest path from n_1 to n_2 that straightly passes through the congested area, and the path that goes along the boundary of the congested area in clockwise direction, which improved the communication performance.

In case of 500 [Kbps] interference shown in **Fig. 8** (a)(b)(c), we also see that the proposed method clearly improves the performance, and the saturation in the conventional method occurs far earlier than the 300 [Kbps] case. We notice that the delivery ratio of the proposed method declines in the stage of low transmission rate, meaning that the proposed method is still not free from the interference; detecting congestion requires several retransmission of frames, and so packets occasionally go into the collision. Such traffic would be affected significantly by the interference at the AP. Also in this case, we observed that the two distinct paths mentioned above worked to obtain load-balancing effects.

In Scenario 2, we see that the proposed method improves the communication performance in the multi-flow scenario. Note that it is natural that the performance gain is smaller than Scenario 1 because clear congestion area does not exist. As for the communication paths, we observed that packets were traveling large area of the field regardless of the original shortest paths, meaning



(a) Throughput

(b) Delivery Ratio

(c) Delivery Delay

Fig. 9 Results of Scenario 2.

that many nodes forward packets with both next-hops, i.e., primary and detour next-hops, so that many distinct paths are used to forward packets for load balancing as well as avoiding congestion. We conclude that the load balancing function provided by the proposed multi-path scheme actually works to improve the communication performance. However, the gain was not so large compared to the Scenario 1 because many paths are crossing and interfere with one another due to the property of the current PHY and MAC protocols.

5. Discussion

From the evaluation, we see that the proposed methods actually provides the detour paths that avoid local congestion at every node, and it works to forward packets along the alternative paths to obtain the gain in communication performance as the load-balancing effects. However, in fact, the gain was quite limited because multiple paths used in our scenarios interfere with one another, and degrade the whole performance of communications. Note that this phenomenon has been already observed in the past multi-path studies such as Refs. [5], [6], which is not the matter of routing paths, but also the matter of PHY and MAC protocols. In other words, unless PHY and MAC protocols are significantly improved to reduce the interference among flows, it would be hard for wireless multi-path routing schemes to obtain the gain from multi-path load balancing.

In contrast, however, we would still emphasize that building effective multiple paths would include an important contribution for the future network technologies. As long as using wireless communications, avoiding congested area instead of single link/node would be a valid strategy from the property of radio waves. In this study, we demonstrated that our distributed algorithm to build detour paths actually works to provide alternative paths at each node on top of a link-state routing protocol. Presenting an algorithm to build feasible detour paths for MANETs would be an important contribution. On the other hand, we dare to say that inventing a new PHY or MAC algorithms that avoid collision in wireless multi-hop networks is a tough task, consequently we would have to wait for a considerable time to solve the problem. One possibility is to use slotted CSMA such as CATBS [13] and IEEE802.15.4 [14]. However, since slotted CSMA is mostly used in sensor networks or mesh networks, we require improvement to apply them into MANETs, which is more dynamic networks. Multi-path routing schemes including this work will be rather important when the lower-layer protocols are improved and achieves interference-free communications among different paths.

6. Conclusion

In this paper, we proposed a distributed algorithm to build detour paths that enable packets in face of congestion to avoid the congested area. Our algorithm runs on each node and computes the detour table. In combination with the additional flags on the packet header, our scheme forwards packets along the detour paths that avoid the congested area. Since the detour table is computed from the 2-hop neighbor information, the proposed machanism can be implemented easily on top of traditional link-state routing protocols such as well-known OLSR, and the time complexity of the additional algorithm is sufficiently low for the practical deployment.

Through the simulation results, we confirmed that the proposed algorithm actually provides the detour paths that avoid the congested area, which is the 1-hop region centered by the next-next-hop node along the shortest path to the destination. In addition, we confirmed that the detour paths effectively functions as a load balancing scheme over the current CSMA-based PHY and MAC framework, in which congestion is detected via retransmission count of IEEE802.11 frames and activate detour paths. However, we also found that, although the proposed scheme works to improve the communication performance, the gain coming from the load balancing function is limited. This is due to heavy interference among flows, where the impact of interference gets worse in general when the number of flows are increased. When a new framework that overcomes this problem is available in the future, multi-path schemes proposed for wireless mesh networks will work more effectively. At that time, our contribution, building detour paths available for every packet that faces congestion, would also perform better and plays an important role on the performance of wireless multi-hop networks.

One of the interesting future work is to consider obstacles in the field. If obstacles such as buildings exist, the detour paths along the circle boundary may not exist. Although our scheme forwards packets into the shortest paths when no detour path is available, it clearly prevents our method to work effectively and improve the communication performance. Extending our scheme for more flexible selection of next hops that avoid congestion even under presence of obstacles such as urban scenario, is one of the next important tasks.

References

[1] Clausen, T. and Jacquet, P.: Optimized Link State Routing Protocol (OLSR), IETF RFC 3626 (2003).

[2] Perkins, C., Belding-Royer, E. and Das, S.: Ad hoc On-Demand Distance Vector (AODV) Routing, IETF RFC3561 (2003).

[3] 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).

[4] Tsai, J. and Moors, T.: A Review of Multipath Routing Protocols From Wireless Ad Hoc to Mesh Network, *Proc. ACoRN Early Career Researcher Workshop on Wireless Multihop Networking*, Australia (2006).

[5] Yi, J., Adnane, A., David, S. and Parrein, B.: Multipath Optimized Link State Routing for Mobile Ad-hoc Networks, *Ad Hoc Networks*, Vol.9, No.1, pp.28–47 (2011).

[6] Marina, K.K. and Das, S.R.: On-demand Multipath Distance Vector Routing in Ad hoc Networks, *Proc. IEEE ICNP*, pp.14–23 (2001).

[7] Lee, S.J. and Gerla, M.: AODV-BR: Backup Routing in Ad hoc Networks, *Proc. IEEE WCNC2000* (2000).

[8] Lee, S.J. and Gerla, M.: Split Multipath Routing with Maximally Disjoint Paths in Ad Hoc Networks, IEEE ICC2001 (2001).

[9] Zhong, Z., Nelakuditi, S., Yu, Y., Lee, S., Wang, J. and Chuah, C.N.: Failure inferencing based fast rerouting for handling transient link and node failures, *Proc. IEEE Global Internet* (Mar. 2005).

[10] Shand, M., Bryand, S. and Previdi, S.: IP Fast Reroute Using Not-via Addresses, draft-ietf-rtgwg-ipfrr-notvia-addresses-04.txt (2009).

[11] Hansen, A.F., Egeland, G. and Engelstad, P.: Could Proactive Link-State Routed Wireless Networks Benefit from Local Fast Reroute?, *6th Annual Conference on Communication Networks and Services Research* (*CNSR2008*), pp.453–462 (2008).

[12] Network Simulator Scenargie, available from ⟨https://www.spacetime-eng.com/en/products⟩.

[13] Yoshihiro, T. and Nishimae, T.: Practical Fast Scheduling and Routing over Slotted CSMA for Wireless MeshNetworks, *Proc. IEEE/ACM International Symposium on Quality of Service* (*IWQoS2016*) (2016).

[14] IEEE 802.15.4b standard, Wireless Medium Access Control and Physical Layer Specification for Low Rate Wireless Personal Area Networks (2006).

Kiyotaka Kaji received his B.E. and M.E. degrees from Wakayama University in 2013 and 2015, respectively. He is interested in wireless ad-hoc networking and routing issues. He is currently with NEC Solution Innovators, Ltd.

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.