DISJOINT PATH ROUTING FOR MULTI-CHANNEL MULTI-INTERFACE WIRELESS MESH NETWORK

Takeshi Ikenaga[1], Koji Tsubouchi[1], Daiki Nobayashi[1], Yutaka Fukuda[2]

[1]Graduate School of Engineering, Kyushu Institute of Technology, Fukuoka, Japan
ike@ecs.kyutech.ac.jp, nova@net.ecs.kyutech.ac.jp
[2]Information Science Center, Kyushu Institute of Technology, Fukuoka, Japan
fukuda@isc.kyutech.ac.jp

ABSTRACT

Dynamic Channel Switching System (DCSS) has been proposed as a solution to improve the capacity of Wireless Mesh Networks (WMNs) which is compliant with IEEE 802.11 standards. DCSS divides wireless interfaces into data receiving and data sending interfaces. The receiving interfaces stay on specified channels, while the sending interfaces can be switched as necessary. A node sends the data by switching the sending interface to the channel assigned to the receiving interface of the communicating node. However, outward and return paths routed by conventional routing algorithms are generally the same. As a result, intermediate nodes have to switch their interface channel frequently and compete with other nodes to use receiver's channel. This leads to significant degradation in their throughput. Therefore, this paper proposes a disjoint path routing scheme in order to prevent the frequent interface switching and improve the throughput performance in the WMN. Simulations revealed that the proposed routing scheme can produce excellent performance compared with the conventional routing schemes.

KEYWORDS

Wireless Mesh Network, IEEE 802.11, Interface Switching, Disjoint Paths, Routing

1. INTRODUCTION

Wireless network capacity can easily be extended with little cost if access points are connected to each other using wireless links. Such networks are called wireless mesh networks (WMNs). WMNs have recently attracted attention according with the standardization of WMNs in IEEE 802.11 TGs [1],[2]. WMNs can be roughly divided into mesh routers and mesh clients. Mesh routers are statically placed, while mesh clients have mobility. Mesh routers form the backbone of the WMN and provide connectivity to the WMN or external networks for the mesh clients. In this paper, a mesh router is denoted as a node, unless explicitly stated otherwise.

In WMN, due to the exposed terminal problem [3], the performance is significantly degraded if a node have only single interface. In order to improve the performance, the node can use multiple wireless interfaces [4]. This approach has already been in practical use [5]-[7]. Some researchers also have proposed a system where the nodes have multiple interfaces and fully use the available channels by switching the interface to another channel according to the next hop nodes [7]-[9]. We denote this system as dynamic channel switching system (DCSS). DCSS divides wireless interfaces into data receiving and data sending interfaces. The receiving interfaces stay on the specified channels, while the sending interfaces can be switched as necessary. A node sends data by switching the sending interface to the channel assigned to the receiving interface of the communicating node. DCSS can provide good performance, leading to all channels being used with fewer numbers of interfaces.

DOI : 10.5121/ijcnc.2011.3211

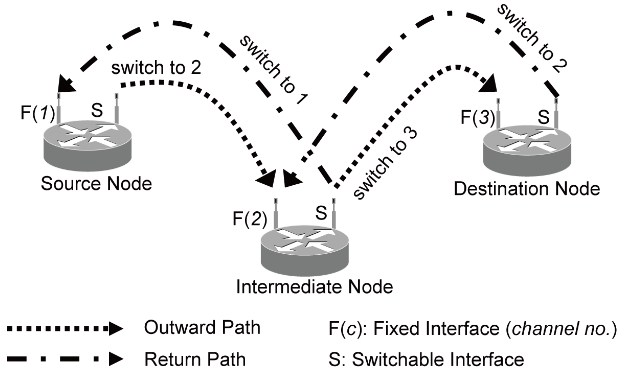


Figure 1. Communication on DCSS.

In general, conventional routing schemes create the same routes for both the outward and return path in WMNs. Thus, channel interference occurs on the receiving interface of the intermediate nodes due to sharing their receiving interfaces. In addition, the intermediate nodes frequently have to switch the interface from one channel to another because the sending buffers in the intermediate nodes have both source and destination packets. Switching an interface from one channel to another incurs a non-negligible delay, which can lead to significant performance degradation. Thus, an effective cooperation between routing and DCSS is required in order to achieve an efficient share of wireless resources.

Therefore, we propose a new routing scheme which uses node disjoint paths that do not include the same node between the outward and return path. The performance of this scheme was evaluated using simulations and compared with the conventional routing schemes. The results revealed that it can achieve the excellent performance when the number of concurrent flows in the network is relatively small, and attain sufficiently comparable performance to that of conventional schemes when the network is congested.

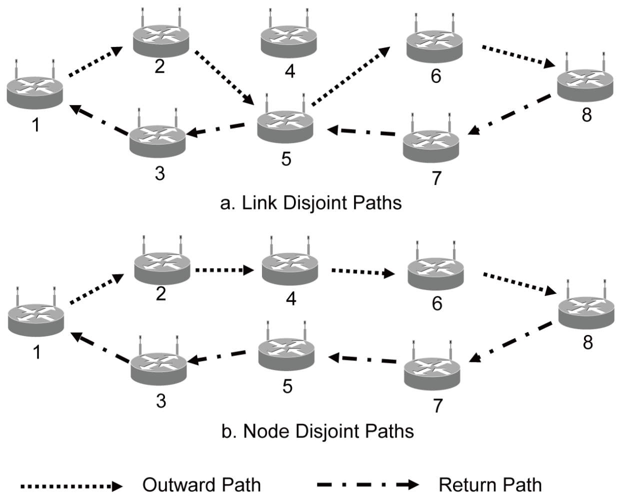
The rest of the paper is organized as follows. Section 2 describes the DCSS and its problems, while section 3 introduces the routing schemes for disjoint paths which could address the problem in DCSS. Section 4 describes the proposed scheme, while section 5 describes the results of the simulation work. Section 6 presents the conclusion.

2. DCSS AND ITS PROBLEMS

This section presents an overview of the DCSS and its problems. DCSS runs using Distributed Coordination Function (DCF), which is the basic access procedure, and uses Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). DCSS is a system which can enhance network throughput by fully using the multiple channels efficiently on the condition the nodes have fewer interfaces than available channels in the WMN.

2.1. Communication procedure of DCSS

DCSS divides the wireless interfaces into "fixed" and "switchable" interfaces. Fixed interfaces stay on the specified "fixed channels", while switchable interfaces can be switched as necessary. Nodes send data by switching their switchable interfaces to the fixed channel of the communicating node. Figure 1. shows an example of communication using DCSS. By distributing all available channels and assigning the channels to the fixed interfaces in the WMN, all the channels can be used, while the switchable interface can be used to maintain connectivity. We assume that each node has at least two interfaces, i.e., one fixed channel and



a. Link Disjoint Paths

b. Node Disjoint Paths

Figure 2. Link disjoint and node disjoint paths.

one switchable channel. For simplicity in the discussion as well, each node will be assumed to have one fixed channel and one switchable channel.

Conventional routing protocols can run without any protocol modifications on DCSS, even if they support only single interface, because DCSS provides an abstract wireless interface to the IP layer. The details of this mechanism are discussed in [8].

2.2. Problems of interface switching delay

A node which employs DCSS switches from one channel to another according to the next hop node when forwarding a packet. When the node forwards a unicast packet but the channel assigned to the switchable interface is different from the fixed channel of communicating node, the forwarding node needs to change the channel of the switchable interface before sending the packet. The node also needs to switch the interface to every channel when it sends a broadcast packet. However, switching an interface from one channel to another incurs some delay, which is known as *switching delay*, which may not be negligible. Current estimates for switching delay are in range between 150 and 200 μs when switching between channels in the same frequency band with commodity IEEE 802.11 hardware, and the delay are more than 5 ms when we take into account the software overhead [8],[11],[12]. This value is relatively large and non-negligible compared to the time of Distributed Inter Frame Space (DIFS), which is the minimum waiting time before sending out a frame, e.g., 34 μs in IEEE 802.11a.

Furthermore, conventional routing schemes create the same routes for both the outward and return paths as shown in Figure 1, which makes the intermediate nodes a bottleneck in two points. One is that when intermediate nodes relay source and destination packets, they share their fixed interfaces. For this reason, channel interference occurs on the fixed interface of the intermediate nodes. The other is that the intermediate nodes frequently switch the interface from one channel to another because the sending buffers in the intermediate nodes have both source and destination packets. Since interface switching incurs non-negligible delay, frequent interface switching results in a large overhead. However, this problem can be addressed by considering the routing scheme which creates disjoint paths in WMN.

3. ROUTING SCHEMES USING DISJOINT PATHS

Disjoint paths can be divided into link disjoint paths and node disjoint paths as shown in Figure 2. In link disjoint paths, the outward path uses different links from the links that are used by the return path. Node disjoint paths not only have different links, but they also do not share any intermediate nodes. When nodes communicate using DCSS as shown in Figure 2, node 5 becomes the bottleneck if link disjoint paths are used, while this problem does not occur if node disjoint paths are used.

In wired networks, disjoint paths are widely used for the support of Quality of Service (QoS) and the restoration of paths after the link failure. Orda et al. [13] present approximation algorithms with provable performance guarantees for finding two link disjoint paths which incur a small violation of the QoS constraint and whose cost guaranteed to be within a certain factor away from the optimum. Bhandari [14] also proposes the algorithm which calculates both link and node disjoint paths respectively in order to recover the node or link failure. The algorithm can find multiple disjoint paths.

In wireless ad-hoc networks, on-demand routing protocols (e.g. AODV, DSR) are widely used because network topology dynamically changes due to mobility. However, these protocols flood messages when reconstructing routes, then this behavior result in large overhead. Therefore, several schemes which reduce flooding overhead using disjoint paths have been introduced [15],[16]. These schemes find multiple disjoint paths to every destination when flooding route requests (RREQs).

In WMNs, Cordeiro et al. [17] propose a routing scheme to use node disjoint paths by considering network load in a similar fashion in [15],[16]. However, since mesh routers are stationary in WMNs, link failures caused by the change of network topology do not occur frequently. Thus proactive routing protocols, which have less flooding overhead, are considered to be suitable for WMNs.

4. PROPOSED SCHEME

We propose a routing scheme which uses node disjoint paths in order to address the problems when using DCSS in WMN as discussed in section 2. Using node disjoint paths can improve bidirectional communication performance by suppressing interface switching and channel interference in WMN using DCSS. Node disjoint path routing using DCSS in WMN is the first application as far as we know.

The proposed scheme employs source routing which is table driven. Each node updates the link state database, which is same as the general link state routing protocol, and then computes two disjoint paths for every destination. The proposed scheme, however, saves only one route in the computed two disjoint paths. Consequently, the path selection scheme in two disjoint paths becomes important.

The proposed algorithm for finding the node disjoint paths is based on the algorithm for finding the link disjoint paths [13]. We simplify this algorithm by removing the QoS constraint and add the function to find the node disjoint path.

The remainder of this section presents the network model and the overview of the theory of network flows which is used for the proposed algorithm. We then propose the algorithm for finding node disjoint paths. Finally, the path selection scheme in two disjoint paths is presented. The example for the use of the algorithm is described in Appendix A.

4.1. Network model

We represent the network by a directed graph *G(V, E)*, where *V* is the set of nodes and *E* is the set of links. A *(s, t)-path* is a finite sequence of distinct nodes $P=(s=v_0, v_1, \ldots, t=v_n)$, such that, for $0 \leq i \leq n-1$, $(v_i, v_{i+1}) \in E$. Here, $n=|P|$ is the hop count of *P*. A cycle is a path whose source and terminating nodes are identical.

Each link $l \in E$ have a cost c_l. In the proposed scheme, the cost is the hop count, and c_l is always 1. Hence, the cost *C(P)* of *P* is $\sum_{l \in P} c_l = n$.

We consider a flow network to conveniently describe the two disjoint paths as a flow in this section. We assume that for any pair of nodes *u* and *v*, the flow network does not contain two links in the opposite directions *((v, u)* and *(u, v))*. We also restrict the flows to take the value of 0 or 1 for each of the links. Thus, for all $l=(u, v) \in E$, it holds that $f_l \in \{0, 1\}$. Also, each link $l \in E$ for which $f_l=1$ belongs to the flow *f*, and *f* includes all links for which $f_l=1$. The flow can be represented by a set of paths P={$P_1, \ldots, P_k$} and cycles W={$W_1, \ldots, W_x$}. The flow f_l on each link *l* which belongs to a path in P or to a cycle in W is 1, while the flow on any other link is 0. The value of a flow *f* is defined as follows:

$$|f| = \sum_{v:(s,v)\in E} f_{(s,v)} \tag{1}$$

A flow of zero value does not contain any paths. Note that a flow *f* with $|f| = 2$ can be decomposed into two disjoint paths.

Given a network *G* with unit capacities and flow *f*, the residual network *G(f)* is constructed as follows. For each link $(u, v) \in G$ for which $f_{(u,v)}=0$, we add to *G(f)* a link *(u,v)* of the same cost as in *G*. For each link $(u, v) \in G$ for which $f_{(u,v)}=1$, we add to *G(f)* a reverse link *(v, u)* to *G(f)* of zero cost.

4.2. Algorithm for finding node disjoint paths

We summarize the proposed algorithm in Figure 3. The main part of the algorithm is to find two link disjoint paths by the procedure **FindLinkDisjointPaths**, and if these paths are still not node disjoint, then the algorithm constructs node disjoint paths from these paths by the procedure **ImproveFlow**.

The procedure **FindLinkDisjointPaths** identifies a suitable flow *f* between source and destination node *s* and *t* such that $|f| = 2$, and it decompose the flow into two disjoint paths $\hat{P}_1$ and $\hat{P}_2$ using the path augmentation approach [18]. Specifically, the procedure firstly computes a shortest hop path P_1 between *s* and *t*. This path defines a flow $f = P_1$ and $|f| = 1$. The procedure next augments this flow to increase its value to 2. For that purpose, the procedure constructs a residual network *G(f)* imposed by the flow *f*, and computes another shortest hop path P_2 between *s* and *t*. The procedure then augments the flow *f* along P_2, that is, for each link *l* which belongs to P_2, we set $f_l=1$ if $f_l=0$, otherwise set $f_l=0$. The value of the resultant flow *f* is 2. Finally, the procedure decomposes the flow *f* into two paths $\hat{P}_1$ and $\hat{P}_2$. The decomposing algorithm is described in [13].

```
Algorithm for Node Disjoint Paths(G, s, t)
input:
  $G$ - the graph, $s$ - source node, $t$ - destination node
output:
  ( $\hat{P}_1$, $\hat{P}_2$ ) - two node disjoint paths

1. ( $P_1$, $P_2$ ) ← FindLinkDisjointPaths($G, s, t$)
2. $f^0$ ← { $P_1$ , $P_2$ }
3. if $P_1$ and $P_2$ are node disjoint, then return $P_1$ and $P_2$
4. $f$ ← ImproveFlow( $G, f^0$ )
5. Decompose flow $f$ into two paths, $P_1$ and $P_2$
6. Jump to 3.

Procedure FindLinkDisjointPaths ($G, s, t$)
1. Identify path $P_1$ in $G$ by using Dijkstra's Algorithm
2. $f$ ← { $P_1$ }
3. Construct the residual network $G(f)$ of $G$ imposed by $f$ :
4.     Add to $G(f)$ each link in $G$ that doesn't belong to $P_1$
5.     for each link $(u,v) \in P_1$ do
6.         Add a link $(v,u)$ to $G(f)$ with $c_{(v,u)} = 0$
7. Identify path $P_2$ in $G(f)$ by using Dijkstra's Algorithm
8. Augment flow $f$ along path $P_2$ :
9.     for each link $l_{(u,v)} \in P_2$ do
10.        if $f_{(v,u)} = 0$ then
11.            $f_{(u,v)}$ ← 1
12.        else
13.            $f_{(v,u)}$ ← 0
14. Decompose flow $f$ into two paths, $\hat{P}_1$ and $\hat{P}_2$
15. return $\hat{P}_1$ and $\hat{P}_2$

Procedure ImproveFlow ($G, f^0$)
1. $f$ ← $f^0$
2. Construct the residual network $G(f)$ of $G$ imposed by $f$ :
3.     Add to $G(f)$ each link $l$ in $G$ for which $f_l = 0$
4.     for each link $(u,v)$ in $G$ for which $f_{(u,v)} = 1$ do
5.         Add a link $(v,u)$ to $G(f)$ with $c_{(v,u)} = 0$
6. Find the cycle $W$ including a node with crossed paths in $G(f)$ that minimizes $C(W)$
7. Augment flow $f$ along $W$
8. return $f$
```

Figure 3. Algorithm for finding node disjoint paths.

The procedure **ImproveFlow** creates node disjoint paths from link disjoint paths. Specifically, the procedure firstly constructs a new residual network imposed by *f*, and finds a cycle which includes the node with crossed flows. Finally, the procedure augments the flow *f* along this cycle.

There could exit no node disjoint paths due to small topology. In such case, if there exists link disjoint paths, the proposed scheme uses these paths, or the same paths as conventional routing schemes.

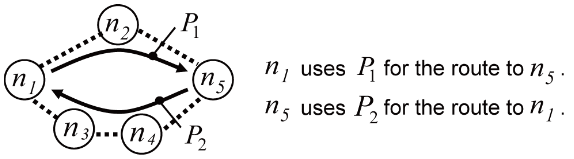
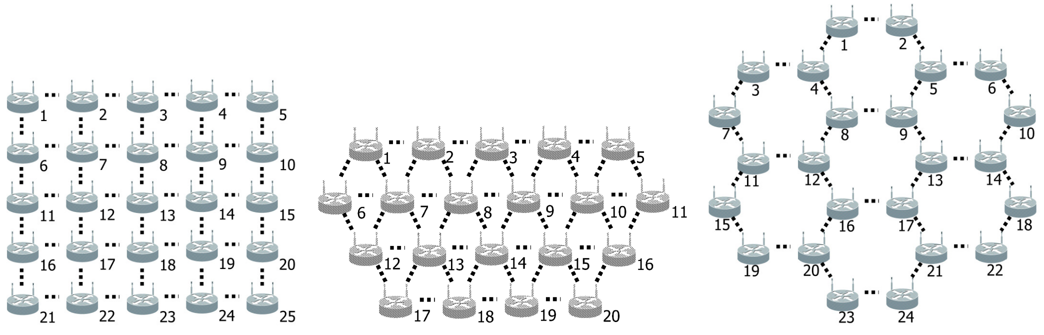

n_2
P_1
n_1
n_5
n_3
n_4
P_2
n_1 uses P_1 for the route to n_5.
n_5 uses P_2 for the route to n_1.

Figure 4. Example of path selection.



(a) Grid topology (b) Triangle topology (c) Hexagon topology

Figure 5. Simulation topologies.

The complexity of the Dijkstra's algorithm generally known is $O(n^2)$. When the proposed algorithm does not call **ImproveFlow**, the complexity of the algorithm still remains in $O(n^2)$. **ImproveFlow** is called when the calculated two paths intersect, and it performs Dijkstra's algorithm for finding a cycle. The worst case is when the two paths intersect at every node on the shorter path. Assuming the hop counts of shorter path are L, the complexity of this case is $L*O(n^2)$.

4.3. Path selection in two disjoint paths

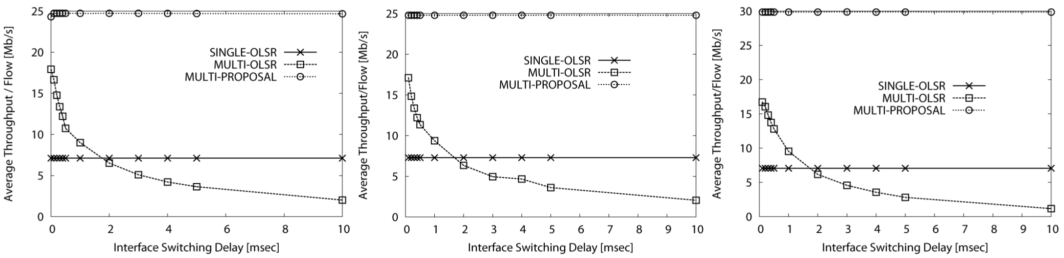
The proposed scheme selects one route in the computed two disjoint paths by using a unique node identifier (ID). Specifically, we assume that node n_x maintains node ID x. In this case, when node n_s finds the route to destination n_t, and if node ID is $s < t$, node n_s computes the disjoint paths P_1 and P_2 from node n_s to n_t and then it stores the path P_1 to the routing table. Otherwise, node $\mathrm{n_s}$ computes the disjoint paths P_1 and P_2 from node n_t to n_s and then it stores P_2 to the routing table as the route from node n_s to n_t.

For example in Figure 4, we consider the case that node n_1 finds the route to n_5. In this case, firstly node n_1 finds the disjoint paths P_1 and P_2 for n_5, then it saves P_1 to the routing table so that node ID 1 is smaller than 5. On the contrary, node n_5 finds P_1 and P_2, then it saves P_2 to the table so that node ID 5 is bigger than 1. In this way, each node can use node disjoint paths autonomously.

5. PERFORMANCE EVALUATION

The proposed scheme was simulated using QualNet 3.9.5 [19]. Figure 5. shows the topologies which we evaluated in the simulation. Each topology has different node density. For each topology, the mesh routers were placed at intervals of 70 m. The dotted lines show wireless links. Any two nodes connected with dotted line can communicate each other. All mesh routers were stationary, since the WMN was considered. Each node used 8 non-overlapping channels.

The transmission rate was fixed at 54 Mb/s. It should be noted that no gateways connected to the wired networks, thus all traffic was within the WMN in our simulations.



(a) Grid topology (b) Triangle topology (c) Hexagon topology

Figure 6. Throughput performance as a function of interface switching delay for each topology.

"MULTI-PROPOSAL" will denote the proposed scheme that works on the node that has two wireless interfaces. "SINGLE/MULTI-OLSR" will denote the OLSR that works on the node that have one or two wireless interfaces. MULTI-PROPOSAL/OLSR employ DCSS. The proposed scheme is compared with SINGLE/MULTI-OLSR.

5.1. Traffic model

First we examine the fundamental performance in order to confirm that the proposed scheme suppresses the interface switching delay by using disjoint paths. In all case, the node 1 send out FTP traffic to destination node for 15 s. The destination node is node 4 in the grid and triangle topologies, and is node 6 in the hexagon topology. MSS is set to 1460 Bytes. Note that FTP traffic has data packets and ACK packets, so this is bidirectional communication.

Next, we deal with a case of multiple flows in the proposed scheme. The source sends out for 15s FTP traffic to a randomly determined destination. MSS is set to 1460 Bytes. This simulation is performed using different source-destination pairs. Then, the performance of the FTP flows of which arrival time is modeled using an exponential distribution was investigated. A randomly selected source-destination pair starts communication. A source sends out the FTP data to the randomly selected destination. One FTP flow is a 10 MBytes transmission.

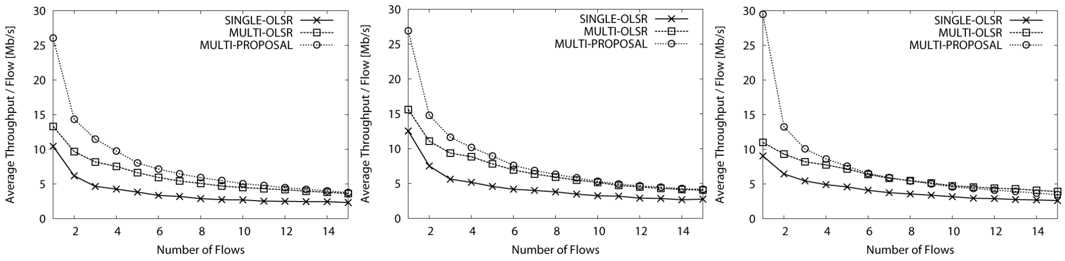
Finally, we examine bidirectional video traffic performance. Video traffic is be represented as constant bit rate traffic and one way traffic sends out by 384 Kb/s by assuming H.264. The communication pairs are randomly chosen.

5.2. Fundamental performance

Figure 6. shows the effects of varying the interface switching delay of the nodes for each topology. In Figure 6(a), MULTI-PROPOSAL constantly shows high average throughput performance despite changing switching delay in all topology. This is because, in the grid topology, MULTI-PROPOSAL uses nodes 1, 2, 3, and 4 as its outward path and nodes 4, 9, 8, 7, 6, and 1 as its return path, and in the triangle topology, MULTI-PROPOSAL uses nodes 1, 2, 3, and 4 as its outward path and nodes 4, 9, 8, 7, and 1 as its return path, and in the hexagon topology, MULTI-PROPOSAL uses nodes 1, 2, 5, and 6 as its outward path and nodes 6, 10, 14, 13, 9, 8, 4, and 1. These paths are node disjoint paths, which implies that interface switching and channel interferences on the fixed interfaces do not occur. On the other hand, for example in the grid topology, MULTI-OLSR uses nodes 1, 2, 3, and 4 as its outward path and nodes 4, 3, 2, and 1 as its return path. In this case, frequent interface switching occurs at nodes 2 and 3, decreasing throughput performance as the switching delay become large. The same effect occurs when using MULTI-OLSR in the other topologies. Therefore, MULTI-PROPOSAL shows a higher throughput performance than MULTI-OLSR even when the switching delay is 0s. From these results, we can say that the proposed scheme can attain excellent performance and is effective for WMN using DCSS.

Table 1. Average hop count / flow for each topology.

Routing Scheme	Average hop count / flow		
	Grid	Triangle	Hexagon
SINGLE/MULTI-OLSR	8.5	7.0	9.0
MULTI-PROPOSAL	8.9	7.3	10.9



(a) Grid topology (b) Triangle topology (c) Hexagon topology

Figure 7. Throughput performance as a function of the number of flows for each topology.

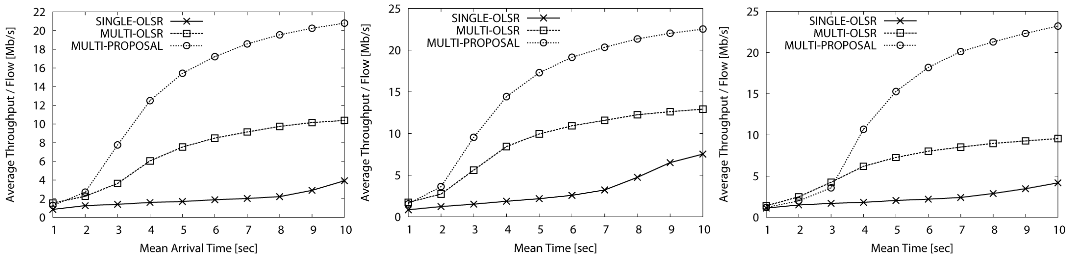
Currently, the interface switching delay including software overhead is more than 5 ms. In the future, this value is expected to decrease, and thus, the impact of this delay on the total overhead will decrease. Therefore, in the next subsection, the effectiveness of the proposed scheme for a small switching delay of 1 ms is presented.

5.3. FTP data transfer

Figure 7. shows the throughput performance as a function of the number of flows concurrently existed in the network, while Table 1 shows the average hop count per round trip path for each topology. From Figure 7. as the number of flows increases, the effect of the proposed scheme, which uses disjoint paths, is reduced. Nevertheless, MULTI-PROPOSAL still obtains the best throughput performance compared to all the other routing schemes in the grid and triangle topology. However in the hexagon topology, throughput performance of MULTI-OLSR is high compared with MULTI-PROPOSAL under high traffic load. Table 1. shows the hop count of each routing scheme for each topology. From Table 1, the hop count of the disjoint paths based on MULTI-PROPOSAL is slightly larger than conventional schemes which use the shortest hop path. The ratio of the hop count of MULTI-PROPOSAL to SINGLE/MULTI-OLSR is the biggest in the hexagon topology compared to those of the other topologies. In general, shortest hop path routing shows the best performance under high traffic load, thus MULTI-OLSR achieves high throughput performance rather than MULTI-PROPOSAL in the hexagon topology.

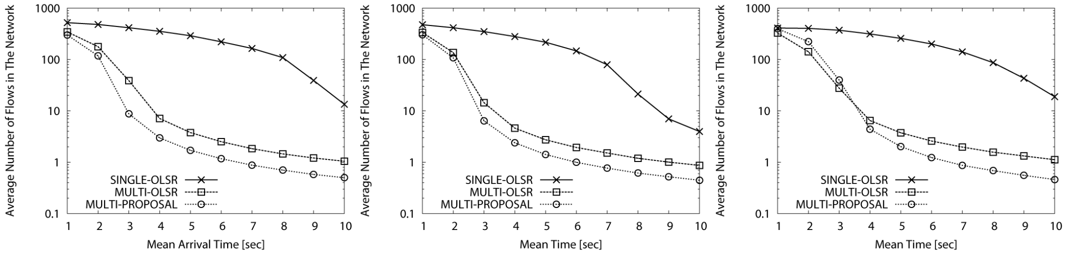
Figure 8. shows the average throughput as a function of flow, while Figure 9. shows the average number of flows that concurrently existed in the network for each topology. As for the horizontal axis of theses figures, the arrival time represents the mean occurrence interval between flows. Therefore, as this interval decreases, the network becomes more congested.

From these figures Figure 8(a). to 9(c). the proposed scheme obtains very good performance when the mean occurrence interval is large, that is, when the number of flows is small.



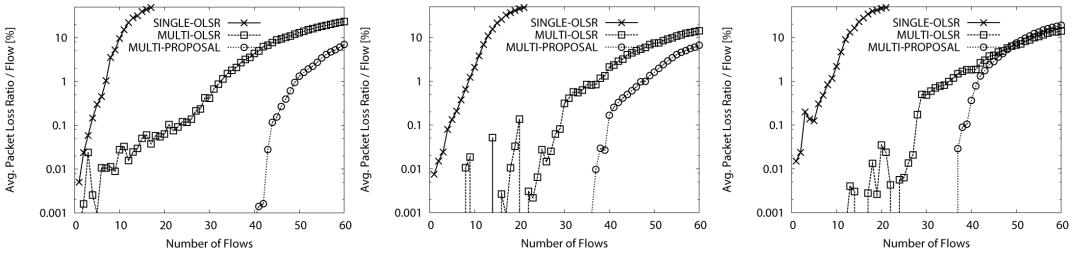
(a) Grid topology (b) Triangle topology (c) Hexagon topology

Figure 8. Throughput performance as a function of mean arrival time for each topology.



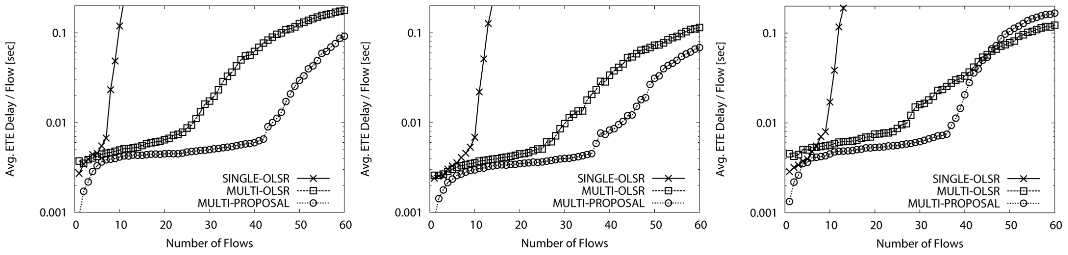
(a) Grid topology (b) Triangle topology (c) Hexagon topology

Figure 9. The number of flows as a function of mean arrival time for each topology.



(a) Grid topology (b) Triangle topology (c) Hexagon topology

Figure 10. Packet loss ratio as a function of the number of flows for each topology.



(a) Grid topology (b) Triangle topology (c) Hexagon topology

Figure 11. End-to-End delay as a function of the number of flows for each topology.

5.4. Bidirectional video transfer

Figure 10. shows the packet loss ratio, while Figure 11. shows end-to-end (ETE) delay, as a function of the number of flows concurrently existed in the network. From these figures Figure 10(a). to 11(c). the proposed scheme obtains the best performance in the schemes. As an example for the grid topology from Figure 10(a), The maximum number of the flows which tolerate 5 % losses, MULTI-PROPOSAL is 57, MULTI-OLSR is 41, and SINGLE-OLSR is 8, thus MULTI-PROPOSAL improves 40 % compared with MULTI-OLSR, and 610 % compared with SINGLE-OLSR. Therefore, we can confirm with the efficiency of the proposed scheme.

6. CONCLUSION AND FUTURE WORKS

The performance of conventional routing schemes in WMN is degraded because these schemes use the same route for the outward and return paths. To solve this problem, we here proposed a new routing scheme that uses node-disjoint paths. The simulation results from grid, triangle mesh, and hexagon topology suggested that when the number of concurrent flows in the network is small, then the proposed routing scheme attained excellent performance. On the other hand, when the network was congested, the performance of the proposed routing scheme was sufficiently comparable to that of conventional routing schemes. Many researchers also have tried to solve the routing and channel assignment for wireless mesh network. For example, Yuan et. al, propose the ROMER[20] as an opportunistic routing scheme for mesh networks. In their scheme, forwarding of each packet is opportunistically adapted to the dynamic channel condition. Thus, as for the future work, we will compare the proposed scheme with other routing or channel assignment schemes which are not based on DCSS.

ACKNOWLEDGEMENTS

This research was partly supported by the Japan Society for the Promotion of Science, Grant-in-Aid for Scientific Research (S) (No. 18100001).

REFERENCES

[1] IEEE Standard 802.11 TGs, http://www.ieee802.org/11/Reports/tgs update.htm.

[2] I. F. Akyildiz, et al., “Wireless mesh networks: a survey,” Computer Networks and ISDN Systems, March 2005.

[3] S. Xu, et al., “Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad Hoc Networks?,” IEEE Communications Magazine, pp.130-137, June 2001.

[4] P. Kyasanur, et al., “Capacity of Multi-Channel Wireless Networks: Impact of Number of Channels and Interfaces,” Proc. of ACM Mobicom, August 2005.

[5] Meshdynamics, inc. http://www.meshdynamics.com/.

[6] Strix Systems, inc. http://www.strixsystems.com/.

[7] P. Kyasanur, et al., “Routing and Link-layer Protocols for Multi-Channel Multi-Interface Ad Hoc Wireless Networks,” SIGMOBILE Mobile Computing and Communications Review, Volume 10, Number 1, pp.31-43, January 2006.

[8] C. Chereddi, et al., “Design and Implementation of a Multi-Channel Multi-Interface Network,” Proc. of REALMAN, May 2006.

[9] L. Loyola, et al., "A New Multi-Channel Mesh Architecture with DCF-Based Inter-AP Communication and Radio-Aware Packet Forwarding for IEEE 802.11-Compliant WLANs," IEICE TRANSACTIONS on Communications Vol.E90-B No.1, pp.78-91, January 2007.

[10] E. Mizoguchi et al., "Design of Access Point with Multiple Interfaces for Multihop Wireless Networks," Technical Report of IEICE, IN2005-220, pp.377-382, March 2006.

[11] Maxim 2.4GHz 802.11b Zero-IF Transceivers. http://datasheets.maxim-ic.com/en/ds/MAX2820-MAX2821A.pdf.

[12] R. Chandra, et al., "MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card," Proc. of IEEE Infocom2004, March 2004.

[13] A. Orda, et. al. "Efficient Algorithms for Computing Disjoint QoS Paths," Proc. of IEEE Infocom2004, 2004.

[14] R. Bhandari, "Optimal Physical Diversity Algorithms and Survivable Networks," Proc. of IEEE ISCC, 1997.

[15] A. Nasipuri, et. al. "Performance of Multipath Routing for On-Demand Protocols in Mobile Ad Hoc Networks," Mobile Networks and Applications, vol.6, 2001.

[16] M. Marina, et. al. "On-demand multipath distance vector routing in ad hoc networks," Proc. of IEEE ICNP2001, November 2001.

[17] C. Cordeiro, et. al. "COPAS: Dynamic Contention-Balancing to Enhance the Performance of TCP over Multihop Wireless Networks," Proc. of the 10th IC3N, October 2002.

[18] R. K. Ahuja, et. al. "Network Flows: Theory, Algorithms, and Applications," Prentice-Hall, 1993.

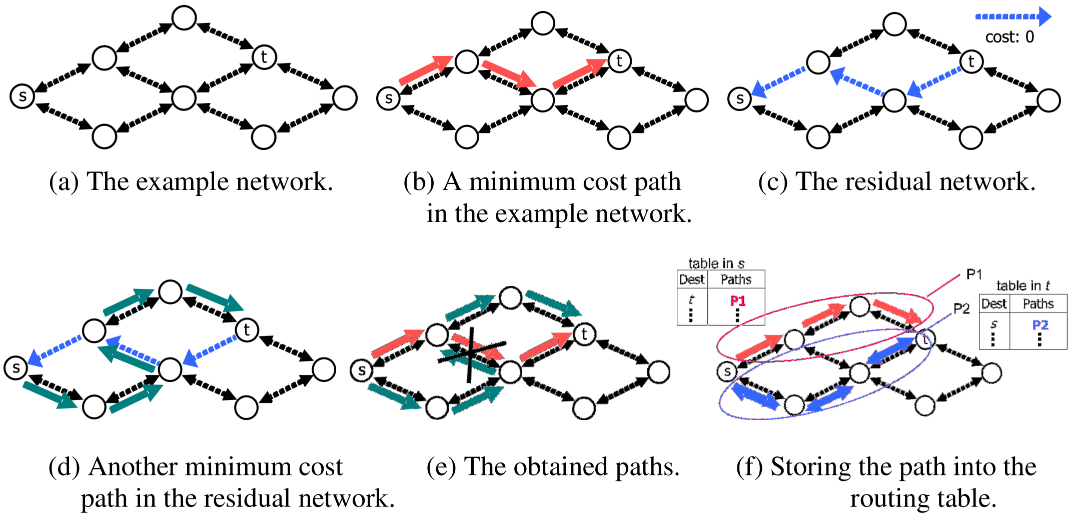
[19] QualNet 3.9.5, http://www.scalable-networks.com.

[20] Y. Yuan, et. al. "ROMER: Resilient opportunistic mesh routing for wireless mesh networks," Proc. of the IEEE Workshop on Wireless Mesh Networks, September 2005.

APPENDIX A: EXAMPLE OF THE USAGE OF THE PROPOSED ALGORITHM

We will explain the algorithm for finding node disjoint paths by using simple example. Figure A.1(a) shows the example network which we will explain here. In this figure, we assume the cost of each link is set to 1, and each link is bidirectional, which is easy to implement the algorithm. When the algorithm tries to find the disjoint paths from node *s* to node *t*, i.e. the case that node number *s* is smaller than *t*, firstly, the algorithm finds a minimum cost path from node *s* to node *t* as shown in Fig. A.1(b). These arrows in Fig. A.1(b) correspond to the flow. Next, the algorithm cuts off the forward direction links along the flow, and it adds the zero cost to the reverse direction links of the flow to construct the residual network imposed by the flow (Fig. A.1(c)). Then, the algorithm finds another minimum cost path in the residual network, and the calculated flows are as shown in Fig. A.1(d). Finally, the algorithm puts together the calculated flows, and in case that bidirectional flow exists, the algorithm deletes it as shown in Fig. A.1(e), and it stores one of the calculated two paths according to the rule explained Section 4.3. Thus, when the algorithm is run in node *s*, node *s* stores P_1 in the routing table. On the other hand, when the algorithm is run in node *t*, node *t* stores the reverse path of P_2 in the routing table as shown in Fig. A.1(f).

When the algorithm only uses previously explained procedure, the calculated paths may be not node disjoint but link disjoint paths. This is for example, if the algorithm calculate a minimum cost path from node *s* to node *t* as shown in Figure A.2(a), the residual network imposed by the flow is calculated as shown in Fig. A.2(b). Then, the algorithm finds a minimum cost path in the residual network as shown in Fig. A.2(c). Finally the algorithm puts together the calculated flows as shown in Fig. A.2(d), whereas these are not node disjoint paths. Therefore, in order to construct node disjoint paths from this flow, the algorithm calculates the residual network imposed by this flow. As a result, the residual network is calculated as shown in Fig. A.2(e). In this network, the algorithm finds a minimum cost cycle which includes the node that intersecting the flow as shown in Fig. A.2(f). Finally, the algorithm puts together the calculated flows, and in case that bidirectional flow exists in a link, the algorithm deletes it as shown in Fig. A.2(g). In this way, even if the paths are not node disjoint, the algorithm can make these paths node disjoint.



(a) The example network.

(b) A minimum cost path in the example network.

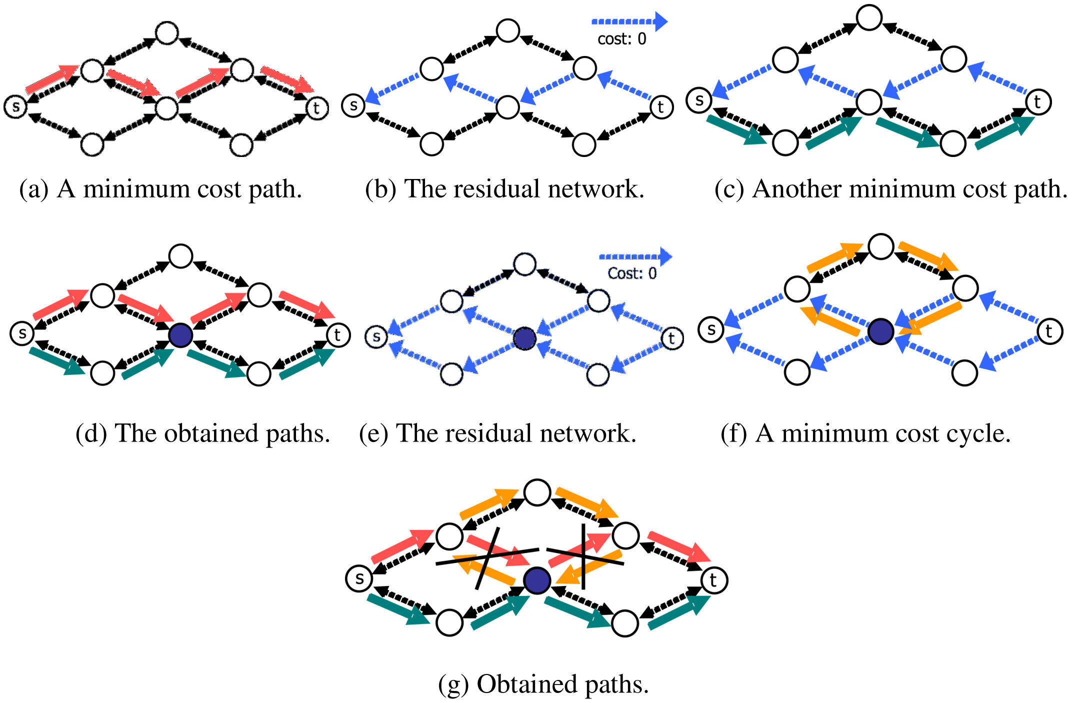
(c) The residual network.

(d) Another minimum cost path in the residual network.

(e) The obtained paths.

(f) Storing the path into the routing table.

Figure A.1. Example of the usage of the algorithm, case 1.



(a) A minimum cost path.

(b) The residual network.

(c) Another minimum cost path.

(d) The obtained paths.

(e) The residual network.

(f) A minimum cost cycle.

(g) Obtained paths.

Figure A.2. Example of the usage of the algorithm, case 2.

Authors

Takeshi Ikenaga received B.E., M.E. and D.E. degrees in computer science from Kyushu Institute of Technology, Iizuka, Japan in 1992, 1994 and 2003, respectively. From 1994 to 1996, he worked at NEC Corporation. From 1996 to 1999, he was an Assistant Professor in the Information Science Center, Nagasaki University, From 1999 to 2004, he was an Assistant Professor in the Department of Computer Science and Electronics, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology, Since March 2004, he has been an Associate Professor in the Department of Electrical, Electronic and Computer Engineering, Faculty of Engineering, Kyushu Institute of Technology. His research interests include performance evaluation of computer networks, wireless LANs and QoS routing. He is a member of the IEEE and IEICE.

Koji Tsubouchi received the B.E. and M.E degrees from Kyushu Institute of Technology, Tobata, Japan in 2006, and 2008. He is currently working at Fujitsu Laboratory Inc. His research interests include wireless network and network architecture.

Daiki Nobayashi received the B.E. and M.E degrees from Kyushu Institute of Technology, Tobata, Japan in 2006, and 2008. He is Graduate student of Graduate School of Engineering, Kyushu Institute of Technology in Japan. His research interests include wireless network, network architecture, and network security.

Yutaka Fukuda received his B.E., M.E., and D.E. in Computer Science from the Kyushu Institute of Technology in Iizuka, Japan in 2000, 2002, and 2005. He has been a Research Associate at the Information Science Center of Kyushu Institute of Technology since October 2003. His research interests include evaluation of the performance of computer networks, wireless networks, and transport protocols. He is a member of the IEEE and IEICE.