A New Routing Scheme Based on Adaptive Selection of Geographic Directions

Zimu Yuan[1,2], Wei Li[2], and Shuhui Yang[3]
[1]University of Chinese Academy of Sciences, China
[2]Institute of Computing Technology, Chinese Academy of Sciences, China
[3]Department of Mathematics, Computer Science, and Statistics, Purdue University Calumet
yuanzimu@ict.ac.cn, liwei@ict.ac.cn, yang246@purdue.edu

Abstract—Geographic routing is recognized as an appealing approach to achieve efficient communications with low computational complexity and space cost. In order to apply this technology in Cyber-Physical Systems (CPSs), a comprehensive consideration must be given to performance issues such as throughput, delay, and load balance. In this paper, we provide a new routing scheme based on forwarding packets to multiple geographic directions. The proposed routing protocols are studied and analyzed theoretically. Theoretical bounds of throughput, delays and space cost are presented. Simulations show that our method performs more efficiently than traditional geographic routing schemes in terms of throughput, delay, and load balance with acceptable space cost. Our experiments also verify the tradeoff between performance metrics.

Index Terms—Geographic routing, protocol, cyber-physical systems, multi-direction routing

I. Introduction

With the emergence of Cyber-Physical Systems (CPSs), wired or wireless network becomes an infrastructure to connect embedded and mobile devices [1]. At the same time, the rapid development of localization technologies for indoor or outdoor environments makes it possible to obtain geographic information from mobile devices. *Geographic routing* [2]-[4], which utilizes physical locations of entities to deliver messages, becomes a promising way to provide robustness, energy saving and low cost for large-scale CPS networks.

In most cases, applications in CPSs ask for good performance in terms of throughput, end-to-end delay, load balance, and guaranteed delivery of messages. These issues were not fully studied in previous research of geographic routing. In addition, most of the existing work on geographic routing focuses on only one or two of these issues. However, these performance issues must be considered comprehensively in order to meet rigid requirements of CPS network. In this paper, we provide a new geographic routing scheme that takes into account all of the above performance issues. The purpose of our method is to use acceptable space cost to achieve good performance.

Our strategy is to provide a new geographic routing scheme, which will forward packets to multiple geographic directions (*Multi-Direction Routing, MDR* for short). After assigning multiple directions to a network, nodes are sorted into global sequences in each direction. A delivery guaranteed protocol is proposed to send/receive messages based on the idea of virtual links. Based on this protocol, an efficient adaptive forwarding method called minimum angle policy is presented. The difference between MDR protocol and other existing geographic rouing protocols is presented in Section 2.

We apply theoretical bounds on throughput, end-to-end delay, stability and the space complexity of the proposed routing protocol. From our analysis, we prove that throughput that is achieved by our method is polynomially proportional to the number of directions assigned to a network, while end-to-end delay that is achieved is the opposite. Also, space complexity of the proposed protocol is the same as that of the classical *Shortest Path Routing (SPR)* protocol.

When we compare the performance of our approach with several typical routing schemes, the experimental results show that our method performs much better in terms of throughput, end-to-end delay and load balance. Moreover, our method can achieve good performance with acceptable space cost. In addition, we provide a classification for geographic routing schemes that reveal cost-performance tradeoff in different routing methods. That is, higher space cost may bring more improvements on performance and vice versa. This insight will be helpful for choosing a proper routing scheme when various limitations in CPS applications are considered.

Our study has been organized as follows: Section II gives the related work including a classification on existing geographic routing schemes; Section III presents the details of the routing scheme proposed; Section IV offers theoretical analyses on throughput, end-to-end delay, stability and space cost; Section V compares our approach with other classical routing methods; and our conclusion is in Section VI.

II. Related Work

Manuscript received January 14, 2014; revised July 28, 2014.
This work was supported in part by the National Basic Research (973) Program of China under Grant No. 2011CB302502, 2011CB302803), and the NSF Grant No. 0943479.
Corresponding author email: yang246@purduecal.edu
doi:10.12720/jcm.9.7.521-534

We provide a classification of existing geographic routing schemes to investigate merits of our proposed method. Basically, geographic routing schemes can be classified into two types: protocols with space cost and protocols without space cost. The former type needs to record route information. The later one only needs to know location information of neighbors of a node. On the other hand, according to their forwarding policies, geographic routing schemes can also be classified into adaptive forwarding and static forwarding. The former can dynamically change policies in the forwarding process; while the later one determines the policy once a message has been sent out. Based on the above analysis, we provide a classification in Table I.

Table I: A CLASSIFICATION OF GEOGRAPHIC ROUTING METHODS

Routing Protocols	Forwarding methods	
	Static	Adaptive
With space cost	I. [2, 3, 5, 6, 7, 8, 9, 10, 11]	II. Our work
No space cost	III. [4, 12, 13, 14, 15, 16, 17, 18, 19]	IV. [20, 21, 22, 23, 24]

The Class I includes the works [2], [3], [5]-[11] on the category of face routing, while Class III includes the works [4], [12]-[19] on the category of graph embedding. Both types adopt the idea of greedy forwarding. A well-known problem of this strategy is that packets may get stuck in a local minimum unfortunately when no neighbors are closer to the destination. To solve this problem, face routing planarlizes the network graph and forward packets along a sequence of adjacent faces towards destinations. It is obvious that this approach needs additional storage cost [4]. Graph embedding will assign virtual coordinates to nodes to ensure greedy forwarding always success. By this means, it does not need to record any route information. Though both methods need no or little storage space, their performance suffers from low throughput or long end-to-end delays. Actually, the nature of face routing is to cut some links to achieve guaranteed deliveries, which will seriously decrease the bandwidth capacity. For graph embedding, inconsistency of virtual coordinates and physical locations may bring remarkable transmission delays.

The work of Class IV also adopts routing protocols with no space cost. Typical methods of this class include [20]-[24]. For example, in [22], randomized approximations to throughput optimal routing were studied. The approach has no significant loss in throughput. It also has good load-balancing properties without compromising latency. However, this method uses multiple copies of packets during transmissions to avoid any cost on routing tables. Apparently this will place a heavy burden on network traffic. The feature of zero space cost makes this approach unable to guarantee delivery of messages, which is a must-have function in many circumstances.

We provide a new geographic routing scheme, which belongs to Class II. Compared with the work of Class I and III, our method can achieve better throughput, end-to-end delay and load balance with acceptable space cost. The extra space cost also enables our method provide guaranteed delivery of messages, which is lacking in routing schemes in Class IV.

III. MULTI-DIRECTION ROUTING SCHEME

In this section, we will first introduce steps of our proposed routing scheme. Then the formal routing protocol is presented and correctness of this approach is proved.

Similar to geographic routing schemes in [2]-[4], our approach (called Multi-Direction Routing, MDR for short) also uses geographic positions of nodes to deliver packets. However, MDR has remarkable differences from previous geographic routing methods:

A. *Assigning Multiple Directions to a Network*

Firstly, we can assign multiple directions to a network. The direction space of a node is splitted equally by each direction. The bisector of the angle between two neighbor directions is the dividing line for the splitted direction space. In Fig. 3.1, the dashed line between direction 1 and 2 is the dividing line. When a packet is forwarded in an orientation with the smallest angle to a special direction such as direction 1, it is said the packet is forwarded to direction 1.

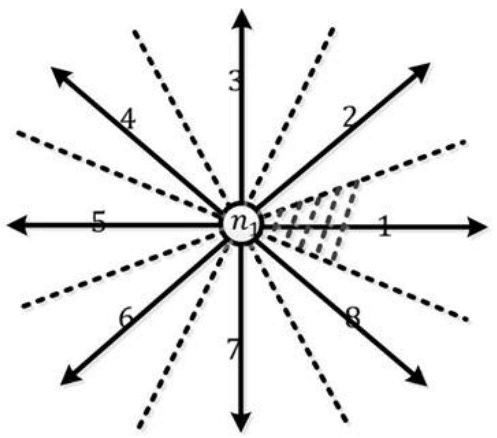


Fig. 3.1. An assignment with 4 IDPs or 8 directions.

A requirement of this assignment is that any direction's inverse direction must be included. We call such a pair of directions as Inverse Direction Pair (IDP for short). In Fig. 3.1, the direction 2 and 6 must be assigned as an IDP at the same time. Therefore, assigning one IDP can achieve mutual communications for any node pair since any node can send packets to its subsequent and front nodes by two inversed directional sequences (e.g. (3-1) and (3-2) in Section 3.2). If a 2D plane is divided equally by IDPs, we say these IDPs construct an even assignment. Fig. 3.1 shows a 2D plane splitted by 4 IDPs equally.

B. *Global Sorting on Geographic Directions*

The nodes are sorted into a sequence according to their coordinate projections on geographic directions. For example, in Fig. 3.2, there are seven nodes n_1, n_2, ···, n_6 and n_7 in a 2D plane. Denoting their projections on the direction *j* as s_1, s_2,···, s_6 and s_7. According to the numerical order (from small to large) of projections, we get a sequence as follows (such a sequence is called *directional sequence*):

$$n_1 < n_2 < n_3 < n_4 < n_5 < n_6 < n_7 \qquad (3\text{-}1)$$

Denoting the inverse direction of the direction *j* as $\overleftarrow{j}$, we get a directional sequence for *j*:

$$n_1 > n_2 > n_3 > n_4 > n_5 > n_6 > n_7 \qquad (3\text{-}2)$$

The direction *j* and its inverse direction $\overleftarrow{j}$ compose an IDP.

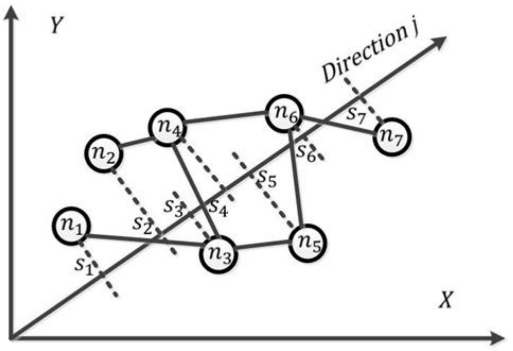


Fig. 3.2. An example of multi-direction routing

If any two adjoined nodes in a directional sequence (e.g. n_3 and n_4, or n_5 and n_6 in Fig. 3.2) are directly linked, it is obvious that any node can send packets to any of its subsequent nodes if all nodes relay packets to their direct subsequent nodes. However, if the adjoined nodes are not directly linked (e.g. n_1 and n_2 in Fig. 3.2), we will construct a *virtual link* (i.e. finding a path between these two nodes) by which a packet can be delivered between the above two adjoined nodes. In fact, for n_1 and n_2, a feasible path does exist, that is, from n_1 to n_3, to n_4 and to n_2. Then this path is regarded, as a virtual link from n_1 to n_2 and it will be recorded in n_1, n_3 and n_4 respectively. According to whether two adjoined nodes are directly linked or not, a node will forward a packet along the virtual link if such a link exists; otherwise, the node will forward the packet to its direct subsequent node. In Fig. 3.2, n_3 records the virtual link from n_1 to n_2. When n_4 receives a packet whose destination is n_2, it forwards the packet to n_2 but not its subsequent node n_5. Nevertheless, for a packet from n_5 to n_7, n_6 will forward it to n_7 without the aid of virtual links since two adjoined node pair n_5 and n_6, n_6 and n_7 are directly linked.

We formally present the sorting algorithm. Assume node n_i establishs its virtual links to nodes in local area. Suppose there are total *N* nodes (including n_i) in this local area and *M* directions or $m/2$ IDPs. By the order of their projections on the direction *j*, we can construct a sequence seq_i of all nodes in a numerical order from small to large according to the distance between projections and the origin. For n_i, we denote the position of its projection on the direction *j* as $rank(i, j)$ and its neighbors (nodes that directly link to n_i) as $nbor(i)$. For $n_k \in nbor(i)$ and any $n_l \in nbor(i)$, if there has $\{n_l \mid rank(i, j) < rank(l, j) < rank(k, j)\} = \varnothing$, we call n_k is the nearest neighbor of n_i in the *j*th direction. Then, the format of the routing table entry in n_i is followed by

$$(< dest_1, dest_2, j >, < ndest_1, ndest_2 >) \qquad (3\text{-}3)$$

Here, $ndest_1$ is the next forwarding node from $dest_1$ to $dest_2$ in direction *j*, and $ndest_2$ is the next forwarding node from $dest_2$ to $dest_1$ in direction *j*. By the above notations, we present our sorting protocol in Algorithm 3.1. (The sorting is localized. In fact, a node only needs to search nearby nodes to find its subsequent node in each direction. A node does not need to know the whole directional sequence of the network.)

Algorithm 3.1. Sorting Protocol

```
Input: Physical positions of N nodes and M directions in local area of node $n_i$
Output: The routing table for node $n_i$ in all directions (taken the $j$ th direction as an example)
Begin
1. Exchanging location information with neighbors;
2. If $rank(i, j) = N$ return;
3. Set $rank' = N$;
4. Finding $n_{ki}$, which is the nearest neighbor of $n_i$; if $n_k$ exists, set $rank' = rank(k, j)$.
5. Establishing virtual links from $n_i$ to $n_h$ where
       $rank(h, j) \in [rank(i, j), rank']$;
End
```

We use Fig. 3.2 to explain Algorithm 3.1. We can see that establishment of the routing table for n_1 in direction *j* is conducted. After exchanging location information, n_1 knows its nearest neighbor is n_3 whose projection's position is 3. Since $rank(2, j) = 2 \in [rank(1, j), rank(3, j)]$, according to Step 5 in Algorithm 3.1, the virtual link between n_1 and n_2 must be created and recorded. The routing table entry in n_1 is similar to the following format:

$$(< n_1, n_2, j >, < n_3, null >) \qquad (3\text{-}4)$$

(3-4) indicates that n_3 is the next forwarding node for a message from n_1 to n_2 in direction *j*. Here *null* in (3-4) indicates that the current node n_1 is the final destination from n_2 to n_1 in direction $\overleftarrow{j}$. For n_3, n_4 and n_2, they also need to add corresponding entries to their routing tables.

In addition, the size of local area of node n_i has choices. The local area could be the one-hop neighbor of node n_i or larger with more nodes involved, even the whole network. Considering the case of larger local area, more routing table entries will be recorded and it will aggravate the storage and searching cost of nodes. So we choose smaller local area for the sorting protocol. The smaller local area includes the next ranking nodes in each direction and the transit nodes of the virtual link to these next ranking nodes. For example in Fig. 3.2, in direction *j*, the local area of n_1 includes the next ranking node n_2 and the transit nodes n_3 and n_4. However, there does not always exist the next ranking node in one direction. When n_i is a boundary node, this case could happen. Several literatures [25]-[27] have fully discussed the boundary detection technique. The boundary detection is the minor issue of MDR protocol. We assume the boundary of the network is already known.

C. *Adaptive for Warding by Selecting Directions*

After constructing routing tables for all nodes, we need to design strategies to forward packets. As concerned, the routing algorithm should achieve good performance in capacity. The forwarding policy is designed based on it. In the MDR scheme, a node can calculate the direction constructed by the source, current location and destination of a packet. We call such a direction as *target direction*. Adaptive forwarding will be achieved if an optimal direction is dynamically selected.

Before we present the forwarding method, we need to explain the interference model first. In ad-hoc networks, transmission links with closely spaced location have mutual interference with each other. It forms a transfer constraint between links that only a portion of links can transmit simultaneously, while the other links of interference cannot be used. Typically, interference can be grouped into two types. The first one typically is called *node-exclusive interference model* [28]. It considers that any two links with a common node, that is 1-hop away, cannot transmit simultaneously. The other model mainly considers the MAC technique in IEEE 802.11 protocol. The nodes should exchange the handshake control message (RTS-CTS before transmitting and DATA-ACK when transmitting) between each other [29].

A distributed scheduling model is built based on these interference models. Regard the links that can interfere the transmitting of link *l* as interference set *I*(*l*). Any interference model should discover the set *I*(*l*). This set is considered already known for general purpose in this paper, no matter which specific interference model is applied. The scheduling model in this paper consists of two parts: *distributed link schedule* that optimizes the flow control in all directions and *distributed packet schedule* that optimizes the flow dispatching in all directions.

1) Distributed maximal weight link schedule for all directions

This schedule gives the flow control policy over links for all directions. The schedule focuses on maximizing the throughput capacity as large as possible. For easy understanding, we explain distributed directional link schedule in detailed steps. Then, we will give the illustration and formal description.

Let $l = \overrightarrow{n_i n_k}$ represent the link that node n_i to node n_k in direction *j*. We define Q_l as the queue length of link *l* in node n_i, and C_l as the transmitting data rate capacity of link *l* from node n_i to node n_j. The links in interference set *I*(*l*) and the end nodes of these links construct the local schedule graph $G_{local}(n_i)$ for node n_i. Let $S(n_i)$ denote the state of node n_i, in which, $S(n_i) \in S = \{uncertainy, transmission, non-transmission\}$. At the beginning, all nodes are at the state of *uncertainy*.

The distributed maximal weight link schedule follows these steps:

- If node n_i is in *uncertainy* state, find the maximal weight direction of node n_i, that is, get link *l* in this direction, $l = \overrightarrow{n_s n_d}$, in which $n_d = \arg\max_{n_d \in nbor(n_i)} Q_l C_l$ and n_s=n_i.
- Find whether the link $l = \overrightarrow{n_s n_d}$ is the maximal weight link in its interference set *I*(*l*). If so, node n_i (or n_s) sets its state as *transmission* and in forms this state change to its neighbor nodes. If not, node n_i keeps in *uncertainy* state.
- If node n_i was informed the transmission message and $n_i = n_d$, node n_i sets its state as *transmission* and informs this state change to its neighbor nodes. Else if node n_i was informed and $n_i \neq n_d$, node n_i sets its state as *non-transmission* and informs this state to its neighbor nodes. Otherwise, ignores the message.
- If node n_i was informed the non-transmission state from node n_j and n_i is in *uncertainy* state, node n_i deletes node n_j and the links with n_j as end nodes in its local schedule graph $G_{local}(n_i)$. Node n_i repeats the steps a) and b).
- If node n_i ($n_i = n_s$ or $n_i = n_d$) finishes the transmission, node n_i sets its state as *uncertainy* and informs this state change to its neighbor nodes.
- If node n_i is in *non-transmission* state and was informed a state change of *transmission* to *uncertainy*, node n_i sets its state to *uncertainy* and informs this state to its neighbor nodes.
- If node n_i was informed a state change of *non-transmission* to *uncertainy* from node n_j and has deleted node n_j and the links with n_j as end nodes in its local schedule graph $G_{local}(n_i)$, node n_i re-adds node n_i and the links into graph $G_{local}(n_i)$.

In summary, when a maximal weight link *l* is selected, the two end nodes of link *l* set their states to *transmission*. All neighbor nodes (1-hop away) of the two end nodes are informed and set their state to *non-transmission*. All neighbor nodes (2-hop away) of these 1-hop neighbor nodes are informed and delete $non-transmission$ state nodes in their local schedule graphs G_{local}. Finishing transmission, nodes in *transmission* and *non-transmission* go back to the state *uncertainy* and nodes re-add these deleted nodes and links in graph G_{local}. In Fig. 3.3, we assume the links are duplex and l_{13} is the maximal link in $I(l_{13})$, in which $I(l_{13})=\{l_{13},l_{24},l_{34},l_{35},l_{46},l_{56}\}$. Let $V(l_{13})=\{n_1,n_3,n_4,n_5\}$. The local graph $G_{local}(n_1)=V(l_{13})\cup I(l_{13})$, that is, the nodes within 2-hop and the links within 2-hop from the links such as l_{13} with n_1 as an end node. When link l_{13} transmits from n_1 to n_3, the node n_2 and n_6 should remove the nodes and links in $G_{local}(n_1)$ from their local schedule graph. When link l_{13} finish transmitting, the node n_2 and n_6 re-add these deleted nodes and links.

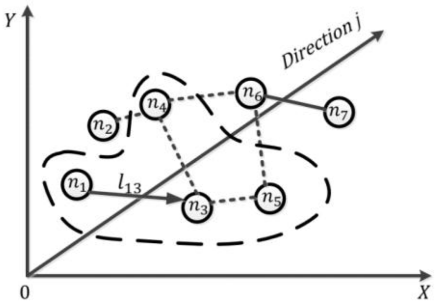


Fig. 3.3. An example of link schedule

The formal description is presented in Algorithm 3.2.

Algorithm 3.2. Link Schedule

```
Input: local schedule graph $G_{local}(n_1)$, queue length $Q$ and capacity $C$ of each link for node $n_i$
Output: link $l$ scheduled
Begin
1. Set $L(n)=\{l \mid l=nn'(nn')\}$
2. Find the link $l=\overrightarrow{n_i n_d}$, $n_d=\arg\max_{n_d\in nbor(n_i)} Q_l C_l$
3. If $l=\arg\max_{l'\in I(l)} Q_{l'} C_{l'}$
4.   set state $S(n_i)$=transmission, notify all neighbor nodes $n$, $n\in nbor(n_i)$
5. If neighbor node $n=n_d$
6.     $n$ set state $S(n)$=transmission, notify all neighbor $n'$, $n'\in nbor(n)$ and all $n'$ set state $S(n')$=non-transmission
7. If neighbor node $n\neq n_d$
8.     $n$ set state $S(n')$=non-transmission, notify all neighbor $n'$, $n'\in nbor(n)$ and all $n'$ set its local schedule graph $G_{local}(n')=G_{local}(n')-\ n-L(n)$.
9. When link $l$ finish transmission
10.  $n$ with $S(n)$=transmission or $S(n)$= non-transmission set its state $S(n)$=uncertainty.
11.  $n$ with $S(n)$=non-transmission notify all neighbor $n'$, $n'\in nbor(n)$ and all $n'$ set its local schedule graph
       $G_{local}(n')=G_{local}(n')+\ n+L(n)$
End
```

2) *Distributed minimum angle based packet schedule for all directions*

Distributed packet schedule optimizes the flow dispatching in all directions. We focus on an adaptive forwarding policy called *minimum angle policy*. At the source node, it is easy to calculate the target direction since locations of the source and destination are known. The target direction will be compared with all directions assigned to the network. Then a closest direction (with the minimum angle) will be selected. After choosing a direction, the forwarding process is accomplished with the aid of routing tables. For example, in Fig. 3.2 or Fig. 3.3, when forwarding a packet from n_1 to n_2 in direction j, we know that the nearest neighbor of n_1 is n_3. Since $rank(n_2,j)\ \in[rank(n_1,j),rank(n_3,j)]$ exists, there has virtual links from n_1 to n_2. When forwarding packets from n_5 to n_7 in direction j, since n_6 is the nearest neighbor of n_5, so n_5 directly forwards packets to n_6 with no need of routing tables. In our method, all relay nodes will use the above procedure until packets reach their destinations.

If there are overmuch packets in direction j for a node n_i, the schedule applied will alternate the direction for some packets to smooth the flow. Let T_l ($0\le T_l\le 1$) be the alternating direction threshold for link l, that is, if the queue length of link l exceeds the threshold of link capacity, $Q_l\ge T_l C_l$, the packet schedule alternates the direction of some packets and send them to other links with surplus capacity.

Assuming packet p is sent from source node s_0 to destination node d_0. Applied with minimum angle based forwarding policy, the direction j_0 is selected to send the packet p. The first item (item 0) recorded in packet p is $<j_0,d_0>$. The other items are recorded in the format of $<j_k,rank(s_0,j_k)>$. There are total *M/2* IDPs (*M* directions) and each IDP has a pair of inverse directions. For each IDP, one direction j is selected with satisfied the condition $rank(d_0,j_k)>\ rank(s_0,j)$. The initial items recorded are shown in Table II. (The cost of adaptive routing is usually shown in three ways. The first is to record routing infos in the packet. The second is to record adaptive routing paths in nodes. The third is to send several copies of original packet to different links. We use the first way that regards adaptive forwarding has closer relation with a special packet itself in direction chose than a node.)

TABLE II: INITIAL ITEM TABLE IN PACKET *P*

No.	Item
0	$<j_0,d_0>$
1	$<j_k,rank(s_0,j_k)>$
…	…
M/2	$<j_{M/2},rank(s_0,j_{M/2})>$

At first, the packet p is delivered along direction j_0. The direction may change during routing. Assume packet p reachs node n_i through direction j_k, node n_i sets the set

$S=\{j \mid rank(d_0,j)>rank(s_0,j)\}$. The distributed minimum angle based packet schedule follows these steps:

- Node n_i finds the minimum angle to the final destination d_0. That is, to find the vector $\overrightarrow{n_i d_0}$ nearest to which direction in set S. If a direction $j_{k'}$ is with the minimum angle, node n_i checks the link l in direction $j_{k'}$.
- If the queue length $Q_l \ge T_l C_l$, remove direction j_k from S. If $S \ne \varnothing$, back to step a).
- Get the items $< j_k, rank(s_k, j_k) >$ and $< j_{k'}, rank(s_{k'}, j_{k'}) >$ from the item table of packet p . If $rank(s_{k'}, j_{k'}) > rank(n_i, j_{k'})$ or $rank(s_k, j_k) > rank(n_i, j_k)$ or $rank(n_i, j_{k'}) > rank(d_0, j_{k'})$, remove direction $j_{k'}$ from S . If $S \ne \varnothing$, back to step a) (Judging if the condition is met in this step is the key to guarantee any packet can be delivered successfully, we will prove the correctness of the protocol based on it). Else overwrite $< j_k, rank(n_i, j_k) >$ and $< j_{k'}, rank(n_i, j_{k'}) >$ to the item table of packet p , change the direction to $j_{k'}$ and overwrite the item 0 : $< j_k, d >$ to $< j_{k'}, d >$.
- Forward the packet p to the next node following the target direction recorded in item 0.

In summary, when a packet p reach node n_i, the direction j has as possible minimum angle with vector $\overrightarrow{n_i d_0}$. The direction j must meet two conditions. The first one is the selected link l in direction j has the queue length $Q_l \le T_l C_l$. The other one is $rank(n_i, j) > rank(s_k, j)$, in which $rank(s_k, j)$ is recorded in the item $< j, rank(s_k, j) >$.

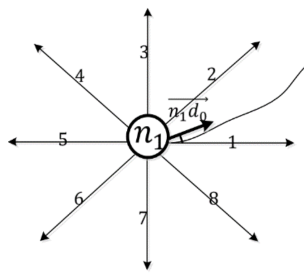


Fig. 3.4. An example of packet schedule

In Fig. 3.4, a packet p is sent from node n_1 to node d_0. At first, the vector $\overrightarrow{n_1 d_0}$ has minimum angle with direction 1, so p is sent in this direction. When p reaches n_i, it alternates to direction 8 by the packet schedule. Along the routing track, p finally reaches destination, node d_0. The formal description is presented in Algorithm 3.3.

Algorithm 3.3. Packet Schedule

```
Input: Packet $p$ (destination $d_0$) and node $n_i$
Output: the direction for $p$ delivery
Begin
1. If $n_i = d_0$ , received
2. Set $S=\{j \mid rank(d_0,j)>rank(n_i,j)\}$
3. Set $j_d = p(0,dir)$ , in which $p(0,dir)$ denotes selecting the direction of item 0 from packet $p$ and
   $p(j,rank)$ denotes selecting rank from item indexed with direction $j$
4. Set $j_{d'} = j_d$
5. While $S \ne \varnothing$
6.     $j_{min} = \arg\min_{j \in S} \theta_j$ , in which $\theta_j$ denote the angle be tween direction $j$ and $\overrightarrow{n_i d_0}$ and select link $l$ in $j_{min}$
7.     If $Q_l \ge T_l C_l$ or $p(j_{\min}, rank) > rank(n_i, j_{\min})$ or $p(j_d, rank) > rank(n_i, j_d)$ or $rank(n_i, j_{\min}) > rank(d_0, j_{\min})$
8.         $S = S - j_{\min}$
9.     Else
10.        Set $p(j_{\min}, rank) = rank(n_i, j_{\min})$ , $p(j_d, rank) = rank(n_i, j_d)$
11.        Set $j_{d'} = j_{min}$
12.        Break
13. If $j_d \ne j_{d'}$
14.     Set $p(0,dir) = j_{d'}$
15. Deliver packet $p$ along direction $p(0,dir)$
End
```

Note that in a distributed system, we do not actually know the global rank of nodes. The physical location of nodes could be record for rank comparison between nodes. Comparison like $rank(s_k, j_k) > rank(n_i, j_k)$ actually is the comparison of perpendicular projection of the physical location of s_k and n_i in direction vector j_k.

D. Summary

In the MDR protocol, node n_i firstly sorts nodes in directions and construct the routing table (Algorithm 3.1). Then, node n_i could use link schedule (Algorithm 3.2) to optimize the flow capacity in all directions and packet schedule (Algorithm 3.3) to smooth the flow in all directions. Link schedule and packet schedule could be carried out in parallel.

The contribution of the MDR protocol mainly lies in that it uses the direction as metric for routing, while current geographical routing protocol uses location as metric. As the metric changes from the location to the direction, the mostly concerned issue that avoiding packets be lost in local minimum needs a new solution. This is the reason why we develop Algorithm 3.1, 3.2 and 3.3. In Section 4 (Proposition 4.1), we will prove that packets with the MDR protocol can avoid local minimum to reach the destination. Also, in Section 4, we will give the theoretical analysis and show how these algorithms of the MDR protocol can improve the performance.

IV. Theoretical Analysis

A. The Model

Fig. 4.1 illustrates the model we used to conduct theoretical results. A random planar network is considered where N nodes are randomly distributed over a unit square whose size is 1×1. An even assignment with M directions or M/2 IDPs is used. The radio range of each node is $r = R\sqrt{(\log N)/N}$, where $R > 1/\sqrt{\pi}$ [30]. The region is further divided in tiles whose length and width are both r (i.e. there are total $u = 1/r^2$ tiles). There are αN (α is a scale factor and $\alpha > 0$) source-destination pairs randomly selected and the routing strategy makes at least δr (δ is a constant factor and $0 < \delta < 1$) progress towards the destination in each forwarding step. Actually, δ is inversely proportional to the total number of M. When more directions are available (i.e. larger M), a better path closer to the target direction could be found, which means δ can take a greater value in average.

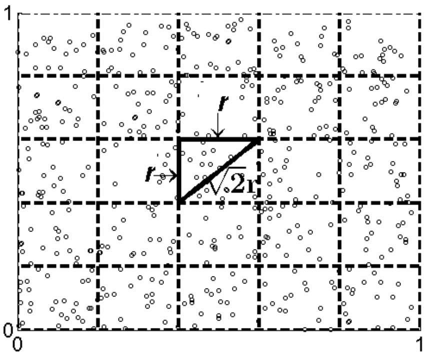


Fig. 4.1. The model

B. The Correctness of The Protocol

First, we will prove the correctness of the MDR protocol that can guarantee delivery of packets.

Proposition 4.1. Any packet can be delivered successfully with the MDR protocol applied.

Proof: Firstly, we prove a packet delivered without alternating direction can be delivered successfully. Then, we explain an alternating direction does not affect the correctness of the MDP protocol.

Without loss of generality, we assume packet p is sent from n_i to n_k (here $i \neq k$) in direction j where $rank(i, j) > rank(k, j)$. Let n_h be the node that the packet locates. At the beginning, we have $n_h = n_i$ ($h \neq k$).

When the packet does not arrive at the destination, we can consider following three cases:

Case 1): If $n_h = nbor(n_k)$, we can directly deliver this packet to node n_k, set $n_h = n_k$.

Case 2): If $rank(h, j) > rank(k, j)$, $Y_i \neq \varnothing$, where $Y_i = \{n_l \mid rank(l, j) < rank(k, j), n_l = nbor(n_h)\}$, we know that there exists some neighbors in direction j, and these nodes are prior to destination n_k in this direction. The packet is delivered to one of node n_l in Y_i, and we set $n_h = n_l$. Here the packet approaching the destination does not skip the destination n_k in this direction.

Case 3): if $rank(h, j) < rank(k, j)$, $Y_i \neq \varnothing$, we know that there must be virtual links that have been established. Then, along the virtual link, the packet is delivered to n_l, which satisfies $rank(h, j) < rank(l, j) < rank(k, j)$. Note that here n_l could be the destination node n_k. Under this condition, the packet approaching the destination does not skip the destination n_k in this direction.

In conclusion, Case 1), 2) and 3) will not skip the destination n_k, thus delivery without alternating direction j is guaranteed.

When alternating direction, packet p can also be delivered successfully. Assume packet p finally routes to destination through direction j'. Let n_i' represent the first node that p starts routing in direction j'. Consider the following two cases:

Case 1): when packet p has arrived at node n_i', it will be routed to the destination n_k without alternating direction. As proved above, packet p will be delivered successfully.

Case 2): when packet p has arrived at node n_i', it will be routed to the destination n_k with alternated directions and finally back to j'. As we look back the condition of alternating directions in Algorithm 3.3. When back to direction j', we have (1) the condition $p(j', rank) > rank(n_i, j')$ must be met. Packet will not back to the nodes of lower rank that has passed by when back to direction j', so the MDR protocol can avoid the local minimum problem. (2) the condition $rank(n_i, j_{\min}) < rank(d_0, j_{\min})$ must be met. This means that the destination d_0 cannot be skipped in direction j'.

As proved, any packet can be delivered successfully with the MDR protocol.

C. Analysis of Throughput

Theorem 4.1. If the MDR protocol and the minimum angle policy are adopted, every source-destination pair can achieve a data rate of $\Theta(j/\sqrt{N \log N})$ simultaneously.

Proof: first, we will prove that it is almost sure that the following inequality stands:

$$Hops \leq \frac{\sqrt{2}}{\delta} H \; (H \sim \Theta(\sqrt{N \log N})) \tag{4-1}$$

$$H = \sqrt{2} aR \frac{\sqrt{N \log N}}{\delta} + \sqrt{10\sqrt{2} aR \log N \frac{\sqrt{N \log N}}{\delta}} \tag{4-2}$$

In (4-1), $Hops$ indicates the total number of forwarding hops of all αN source-destination pairs passing any single tile. For a specific tile, according to the MDR algorithm and the forwarding policy, some source-destination pairs may go through it via one or multiple hops. We use the random variable $X_i^j \in \{0,1\}$ ($1 \leq i \leq 1/r^2$, $1 \leq j \leq aN$) to indicate if the jth path will pass the i th tile. $X_i^j = 1$

means that the *j*th path will pass the i th tile, otherwise not. Then the total number of source-destination pairs passing the i th tile will be $S_i = \sum_{j=1}^{aN} X_i^j$. For the *j*th path, we know that the maximum number of its hops in the *i*th tile will be $h_i = \sqrt{2}/\delta$. Therefore, the total number of forwarding hops in the *i*th tile is upper bounded by the product of h_i and the maximum of S_i. That is

$$Hops \le h_i \cdot \max(S_i) = \frac{\sqrt{2}}{\delta}\max(S_i) \tag{4-3}$$

To obtain (4-1), we first prove that the following inequality has small probability of occurrence:

$$P(Hops > \frac{\sqrt{2}}{\delta}H) \text{ or } P(max(S_i) > H) \tag{4-4}$$

Here P indicates the probability. From the knowledge of statistics, we know that the union bound of the probability is greater than the maximal of the probability. Then, we have

$$P(max(S_i) > H) \le \sum_{i=1}^{1/r^2} P(S_i > H) \tag{4-5}$$

Let $\tilde{X}_i^j$ be the Bernoulli random variable [31] stochastically dominating X_i^j. Then the probability of $P(\tilde{X}_i^j = 1)$ is greater than that of $P(X_i^j = 1)$. Denoting $\sum_{j=1}^{\alpha N} \tilde{X}_i^j$ as $\tilde{S}_i$, then we have

$$P(S_i > H) \text{£} P(\tilde{S}_i > H) \tag{4-6}$$

Putting (4-5) and (4-6) together, we have

$$P\left(max(S_i) > H\right) \le \frac{1}{r^2} P\left(\tilde{S}_i > H\right) \tag{4-7}$$

We instantiate $\tilde{X}_i^j$ and make it dominate X_i^j surely and stochastically. Denoting u_j as the upper bound of the number of tiles passed by the *j*th path. Since the number of jumps required to reach the destination for any route is no more than $\sqrt{2}/(\delta r)$ hops ($\sqrt{2}$ is the maximal length between two nodes), we set $u_i = \sqrt{2}/(\delta r)$. So the probability that a tile may be touched by the *j*th path will be $P_u = u_j / u = \sqrt{2}r/\delta$. Then, considering the random variable $P(\tilde{X}_i^j = 1) = P_u$. Since P_u is the maximum probability of a tile touched by the *j*th path, X_i^j surely stochastically dominates X_i^j at the setting. So, we have $E\left(\tilde{X}^j\right) = \alpha N \cdot P_u$ $E\left(\tilde{X}^j\right) = \ = \sqrt{2}\alpha R\sqrt{NlogN}/\delta$ ($E\left(\tilde{X}^j\right)$ is the expect value of tiles passed by the *j*th path).

Both X_i^j and X_i^j are Bernoulli random variables. X_i^j is independent to $\tilde{X}_l^k$ if $j \ne k$ for any i and l. For the sums of i.i.d. Bernoulli random variables, we have [32]

$$P\left(\tilde{X}^j > (1+\beta)E\left(\tilde{X}^j\right)\right) \le e^{-\beta^2 E\left(\tilde{X}^j\right)/2} \tag{4-8}$$

Set $\beta = \sqrt{10\log N / E\left(\tilde{X}^j\right)}$. Then, (4-8) can be rewritten as

$$P\left(\tilde{X}_i^j > H\right) \le \frac{1}{N^5} \tag{4-9}$$

Combining with (4-9), by summing over all j in (4-7), where $\tilde{S}_i = \sum_{j=1}^{aN} \tilde{X}_i^j$, we have

$$P\left(\max\left(S_i\right) > H\right) \le \frac{1}{R^2 N^4 \log N} \tag{4-10}$$

Therefore $P\left(\max\left(S_i\right) > H\right)$ (or $Hops > \frac{\sqrt{2}}{\delta}H$) has small probability of occurrence. Thus, we prove that it is almost sure that (4-1) will converge.

Next, we will present a bound for throughput. The transmission of a node in a tile may affect transmissions of other nodes in the range influenced in current or neighboring tiles. We assume that at most D neighbor tiles are affected. As refer to the graph coloring problem, all tiles can be colored by D+1 colors. Considering a time interval T with the fixed length, we can divide T into $T/(D+1)$ slots. Rewriting (4-1) as

$$Hops \le A\sqrt{N\log N} \tag{4-11}$$

where A is a finite number (note that $A \sim 1/\delta$ here). Thus, each forwarding hop in any tile can get a transmission time as

$$\frac{1}{A(D+1)\sqrt{N\log N}} \tag{4-12}$$

Therefore, we complete the proof that each source-destination pair can achieve the data rate of $\Theta(1/\sqrt{N\log N})$ simultaneously.

Note. Here $A \sim 1/\delta$ and $M \sim \delta$ means that when more directions are available, the MDR protocol has better throughput.

D. Analysis of End-to-end Delay

Theorem 4.2. If the MDR protocol and the adaptive-based policy are adopted, the delay of any source-destination pair is no more than $\Theta(N)$.

Proof: we know that the total number of hops for a source-destination pair in the square is at most $\sqrt{2}/\delta r$ (the maximal distance between two nodes is $\sqrt{2}$, and the routing strategy progresses δr at least in each forwarding step). As proved in Theorem 4.1, each source-destination pair can achieve a data rate of $\Theta(1/\sqrt{N\log N})$ (see (4-12)) simultaneously. Thus, the delay of each hop is no more than $\Theta\left(\sqrt{NlogN}\right)$. The total delay is no more than $\frac{\sqrt{2}}{\delta r}\Theta\left(\sqrt{N\log N}\right) = \Theta\left(N\right)$.

E. Cost of Routing Tables

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Do not use abbreviations in the title unless they are unavoidable.

Theorem 4.3. In Algorithm 3.1, if node n_1 has at least one neighbor node, then at least $\left\lfloor \frac{|M|}{2} \right\rfloor$ directions need not to memorize any routing table entry.

Proof: it is easy to see that in Fig. 4.2, since directions equally divide a plane, the half-plane in the n_2 side of the dotted line has at least $\left\lfloor \frac{|M|}{2} \right\rfloor$ directions (in Fig. 4.2, the dotted line is perpendicular to line n_1n_2). We can see that, in Fig. 4.2, the projection of n_2 on directions is always large than that of n_1, thus we do not need to store any virtual link between n_1 and n_2 for these directions.

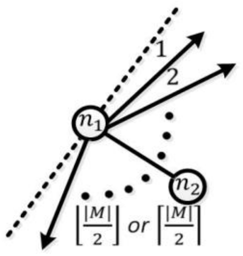

1
2
n_1
n_2
$\left\lfloor \frac{|M|}{2} \right\rfloor$ or $\left\lceil \frac{|M|}{2} \right\rceil$

Fig. 4.2. Illustration of Theorem 4.4

Theorem 4.4. In Algorithm 3.1, an upper bound of the number of routing table entries in a node is $(N-1)\left\lfloor \frac{|M|}{2} \right\rfloor$.

Proof: there are at most $\left\lfloor \frac{|M|}{2} \right\rfloor$ directions needed for a node to record the routing path. Therefore the theorem is proved.

F. Analysis of Stability

Link and direction schedule dispatch the packets and decide the working state of links according to the interference constrains. The schedules are required to ensure the network eventually reaches a stable state. If the queue length Q_l of a link l is finite, the link l is called *stable*. If all links in the network are stable, the network is called *stable*. Furthermore, the network stability means that the loads are averaged to nodes and the load of each node is in scope of tolerance. Next, we prove the stability of MDR protocol by analysis. Then, in Section 5, we give the experiment results of load balance comparison between MDP and other protocols.

Let $\lambda = (\lambda_0, \lambda_1, \ldots, \lambda_{n-1})$ represent the arrival rate vector of the network. The literature [33] defines all arrival rate vectors that a schedule algorithm $\prod$ can keep the network stable with as capacity region Λ_Π, that is, $\Lambda_\Pi = \{\lambda \mid schedule?\Pi\, stable\, with\, \lambda\}$. For a special netwok, the union capacity region of all schedule algorithms is called optimal capacity region Λ_{opt}. If any schedule algorithm can reach Λ_{opt}, the schedule is called the optimal schedule Π_{opt}. As defined, C_l is transmitting capacity of link l. Let r_l denote the workable data rate of link l in the current state. As easy to see, $r_l = C_l$ when scheduled and $r_l = 0$ when not scheduled. Let $r = \{r_0, r_1, \ldots, r_{L-1}\}$ denote the workable data rate vector of total L links in the network. Collection of all workable vectors R can be written as $R = \{r \mid r\ workable\ currently\}$. According to literatures [33] [34], Λ_{opt} can be defined as $\Lambda_{\text{opt}} = \{\lambda \mid \lambda \le r, r \in Conv(R)\}$, in which $Conv(R)$ denotes the convex collection of R. This definition implies that for any arrival data rate vector $\lambda' \in \Lambda_{opt}$, there must exist a workable schedule vector $r' \in Conv(R)$, $\lambda' = r'$.

It is hard to find the optimal solution under the constraint of interference between nodes [35]. To find a sub-optimal solution is a workable way. The capacity region of sub-optimal solution reaches $\gamma(0 < \gamma < 1)$ times of optimal solutions. That is, if a schedule Π can keep the network stable with any arrival data rate vector $\lambda \in \gamma\Lambda_{opt}$, the schedule Π is called a sub-optimal solution. Next, we prove our schedule is a sub-optimal solution.

Lemma 4.1. For any arrival data rate vector $\lambda \in \Lambda_{opt}$, there must exist a random schedule that make the expect value of schedule vector $E\{r\} = \lambda$ (That is, the schedule can process the data with the arrival rate λ and do not increase the queue length of each node in the network).

Proof: Under the definition of Λ_{opt}, for any $\lambda \in \Lambda_{opt}$, there must exist a workable schedule vector $r \in Conv(R)$, $\lambda = r$. According to the Caratheodory thorem (The classical theorem in convex geometry), convex set $Conv(R)$ must have:

$$r = P_0 r_0 + P_1 r_1 + \ldots + P_L r_L \tag{4-13}$$

in (4-13), $r_i \in R$ ($i = 0,1,..,L$), $\sum_{i=0}^{L} P_i = 1$. So a random schedule algorithm exists: randomly select a schedule vector r_i from $\{r_0, r_1, \ldots, r_L\}$ with corresponding probability P_i. This schedule has $\{r\} = \lambda$.

Theorem 4.5. The inequality $\sum_{\Pi_{opt}} Q_l r_l \ge \gamma \sum_{\Pi_{MDR}} Q_l r_l$ ($\gamma \ge 1/\kappa_{max}$) is satisfied. In this inequality, $\kappa_{max} = \max_{l\in[0,L-1]} \kappa(l)$, $\kappa(l)$ denotes the maximal count of links that can simultaneously transmit in interference set $I(l)$.

Proof: set the links set of optimal schedule Π_{opt} as S_{opt}, and Π_{MDR} as S_{MDR}. For $\forall l \in S_{opt}$, $l \notin S_{MDR}$, there exists a link set $E(l) \subseteq S_{MDR}$ and $E(l) \subseteq I(l)$. In $E(l)$, there exists link l' and has $Q_l r_l \le Q_{l'} r_{l'}$ (Recalling $Q_l r_l$ is the weight for link schedule of Π_{MDR} and Q_l / r_l is the weight for direction schedule of Π_{MDR} to smooth the flow) . So for set S_{opt} , there must exists set $S'_{MDR} = \{ l'_i \in S_{MDR}, i = 0,1,...,|S_{opt}|-1 \}$.

In S'_{MDR}, some links are the same. Randomly select a group of same links $l'_0 = l'_1 = \ldots = l'_{k-1}$, totally k. Then, there will be k different links $\{l_0, l_1, \ldots, l_{k-1} \in S_{opt}\}$ and $l_i \in I(l'_i)$ for $i = 0,1,\ldots,k-1$. Obviously, $k \le \kappa_{max}$ as defined. Otherwise, the case that any two links l_i and l_i in S_{opt} are the same will contradict to the optimal schedule itself. Another link l_q with $Q_i r_i \le Q_q r_q$ can be found in this case.

As for $\forall l \in S_{opt}$, $l \notin S_{MDR}$, there exists $Q_l r_l \le Q_{l'} r_{l'}$, $l' \in S_{MDR}$. We can get the inequality:

$$\sum_{i=0, l \in S_{opt}}^{|S_{opt}|-1} Q_{l_i} r_{l_i} \le \sum_{i=0, l' \in S'_{MDR}}^{|S'_{MDR}|-1} Q_{l'_i} r_{l'_i} \tag{4-14}$$

Let S''_{MDR} denote the set that selects distinct links from S'_{MDR} . Obviously, the following inequality can be established:

$$\sum_{i=0, l' \in S'_{MDR}}^{|S'_{MDR}|-1} Q_{l'_i} r_{l'_i} \le \kappa_{max} \sum_{i=0, l' \in S''_{MDR}}^{|S''_{MDR}|-1} Q_{l'_i} r_{l'_i} \tag{4-15}$$

According to the definition, for any $l \in S''_{MDR}$, there is $l \in S_{MDR}$. That is, $S''_{MDR} \in S_{MDR}$. Then, we have:

$$\sum_{i=0, l' \in S''_{MDR}}^{|S''_{MDR}|-1} Q_{l'_i} r_{l'_i} \le \sum_{i=0, l' \in S_{MDR}}^{|S_{MDR}|-1} Q_{l'_i} r_{l'_i} \tag{4-16}$$

Combine (4-14), (4-15) and (4-16), we get the result:

$$\sum_{i=0, l' \in S_{MDR}}^{|S_{MDR}|-1} Q_{l'_i} r_{l'_i} \ge \gamma \sum_{i=0, l \in S_{opt}}^{|S_{opt}|-1} Q_{l_i} r_{l_i} \tag{4-17}$$

In (4-17), $\gamma \ge 1/\kappa_{\max}$.

Theorem 4.6. Π_{MDR} is a sub-optimal schedule. For any arrival data rate λ , if $\lambda + \varepsilon \in \gamma\Lambda$ ($\varepsilon = (\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_{L-1})^T$, $\varepsilon_i > 0, i = 0,1,\ldots,L-1$), Then, Π_{MDR} can guarantee the network stable and has the supremum limit $\lim_{s\to\infty} \sup 1/s \sum_{\varsigma=0}^{s-1} E\{Q_l(\varsigma)\} < B/\varepsilon$ (B is constant vector).

Proof: in state s , any link l has $Q_l(s+1) = \max\{Q_l(s) - r_l(s), 0\} + \lambda_l(s)$. As defined, $r_l(s)$ is the schedule capacity and $\lambda_l(s)$ is the arrival data rate of link l in state s. As easy to verify, there is $Q_l^2(s+1) - Q_l^2(s) \le r_l^2(s) + \lambda_l^2(s) - 2Q_l(s)(r_l(s) - \lambda_l(s))$. Define Lyapunov function [36]: $L(Q(s)) = \sum_{l=0}^{L-1} Q_l^2(s)$. We have the following inequality:

$$L(Q(s+1)) - L(Q(s)) \le \sum_{l=0}^{L-1} (r_l^2(s) + \lambda_l^2(s))$$
$$-2\sum_{l=0}^{L-1} \left(Q_l(s)(r_l(s) - \lambda_l(s)) \right) \tag{4-18}$$

Assume $r_{max} = \max r_l(s_i)$ and $\lambda_{max} = \max \lambda_l(s_i)$ for $l = 0,1,...,L-1$, $s_i = 0,1,...,s$. Then, we have

$$\sum_{l=0}^{L-1} \left(r_l^2(s) + \lambda_l^2(s) \right) \le L\left(r_l(s) + \lambda_l(s) \right) = B \tag{4-19}$$

Define Lyapunov drift function [36] as:

$$\Delta Q(s) = E_{MDR}\left(L(Q(s+1)) - L(Q(s)) \right) \tag{4-20}$$

Combine (4-18) (4-19) and (4-20), we have

$$Q(s) \le B + 2\sum_{l=0}^{L-1} Q_l(s)\lambda_l(s)$$
$$-2\sum_{l=0}^{L-1} E_{MDR}(Q_l(s) r_l(s)) \tag{4-21}$$

Refer to Lemma 4.1, we know if $\lambda(s) + \varepsilon \in \gamma\Lambda$, there exists an optimal random schedule Π_{opt} that have：

$$E_{opt}(r(s)) = (\lambda(s) + \varepsilon)/\gamma \tag{4-22}$$

Also, $E(r(s)) = \sum_{l=0}^{L-1} E(r_l(s))$. According to Theorem 4.5, $\sum_{\Pi_{opt}} Q_l r_l \ge \gamma \sum_{\Pi_{MDR}} Q_l r_l$ ($\gamma \ge 1/\kappa_{max}$), we have

$$E_{MDR}(Q(s) r(s)) \ge \gamma E_{opt}(Q(s) r(s))$$
$$\ge \sum_{l=0}^{L-1} Q_l(s)(\lambda(s) + \varepsilon) \tag{4-23}$$

Substitute (4-23) in (4-21), we get:

$$\Delta Q(s) = B - 2\varepsilon \sum_{l=0}^{L-1} Q_l(s) \tag{4-24}$$

Refer to the Lyapunov stability theorem [36], we get the result:

$$\lim_{s\to\infty} \sup 1/s \sum_{\varsigma=0}^{s-1} E\{Q_l(\varsigma)\} < B/\varepsilon \tag{4-25}$$

Therefore, Theorem 4.6 is proved.

V. Evaluations

In this section, we will evaluate and compare our algorithm (Class II) with *Shortest-Path Routing* (*SPR*) algorithm (Class I), *Hyperbolic Geographic Routing* (*HGR*) algorithm (Class III) [17], and *Randomized Local Load Balancing* (*RandLLB*) algorithm (Class IV) [22]. These algorithms are classical methods belonging to different categories in Table I. We perform the experiments on Matlab.

A. Performance Metrics

1) Throughput

Throughput is evaluated by packet delivery ratio R_T. First, the transmission success rate T_θ (where θ can be SPR, HGR and RandLLB) is defined as $T_\theta = (n_a/n_t)\times 100\%$, in which n_a is the number of packets that arrive at destinations; n_t is the total number of packets sent out. Then we have

$$R_T = \frac{T_{MDR}}{T_\theta}\times 100\% \tag{5-1}$$

2) End-to-end delay

We use delay ratio R_D to present end-to-end delay (This includes possible delays such as queuing delays at the interface, propagation delays and transmission delays). After all packets reached their destinations, we record the delay of each packet and denote the delay of the packet i as d_i. Assuming there are totally P packets generated, the average delay D_θ for all packets is defined as $D_\theta = \frac{1}{P}\sum_i^P d_i D_\theta$. Then we have

$$R_D = \frac{D_{MDR}}{D_\theta}\times 100\% \tag{5-2}$$

3) Load balance

We use load ratio R_L to measure the property of load balance. In a given time period, we record the total number of packets processed by node i as l_i. The average number of packets processed by each node is defined as $E_\theta = \frac{1}{P}\sum_i^P l_i$. Then, the variance of the number of packets processed by each node is defined as $V_\theta = \frac{1}{P}\sum_i^P \left(l_i - E_\theta\right)^2$ which indicates how smoothly packets will distribute on all nodes. When V_θ is smaller, it means that packet flows distribute more smoothly in the network. Then we have

$$R_L = \frac{V_{MDR}}{V_\theta}\times 100\% \tag{5-3}$$

4) Routing table size

We use space ratio R_S to evaluate storage cost (i.e. the routing table size). Denoting the routing table size of different algorithms as S_θ, then we have

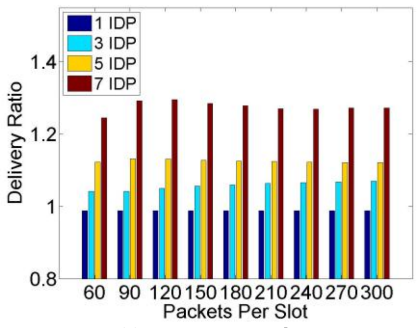
$$R_S = \frac{S_{MDR}}{S_\theta}\times 100\% \tag{5-4}$$

B. Simulation Environment

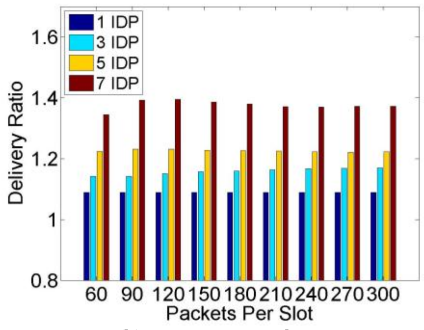
We use following settings in our simulations: in a given region, there are 100 nodes randomly deployed. For each trial of experiments, 10 source-destination pairs of packet flows among nodes are randomly generated. For each flow, the number of generated packets varies from 60 to 300 (in 1000 time slots). For the capability of network nodes, we assume that each node can process 30 packets per slot. For ease of evaluating, we also assume that all nodes have a queue with infinite length and there is no packet loss due to queuing. The simulation will terminate when all packets have been delivered to their destinations. For the direction assignment, we use the even assignment in Fig. 3.1.

C. Comparison of Throughput

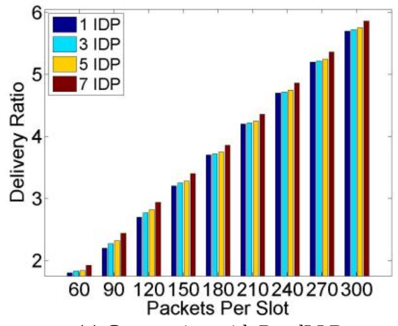
The comparison of throughput is showed in Fig. 5.1 (a), (b), and (c). We can see that: 1) when the number of packets increases, our algorithm performs better than other algorithms if more directions are available (here 3 IDPs are enough); 2) The performance of our algorithm is directly proportional to the number of directions available. That is, the more directions used, the better our algorithm is. Especially, when 7 or more IDPs are selected, our algorithm performs at least 20% better than the SPR algorithm, 30% better than the HGR algorithm. For the reason of packets replication and destination unreachable, delivery ratio of RandLLB is very high.



(a) Comparsion with SPR



(b) Comparsion with HGR

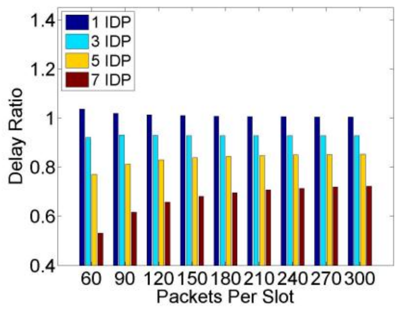


(c) Comparsion with RandLLB

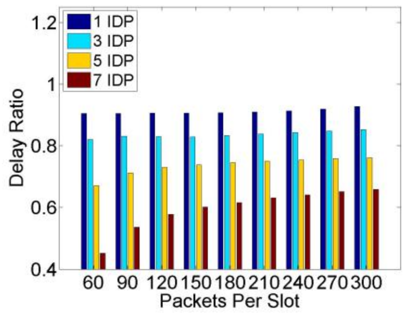
Figure 5.1. Comparison of load balance

D. Comparison of End-to-End Delay

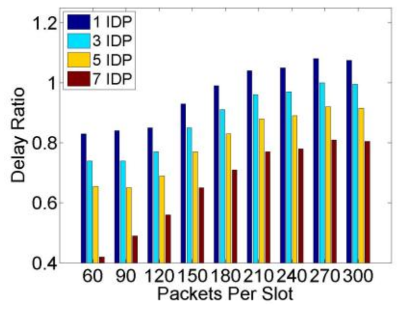
The comparison of end-to-end delay is showed in Fig. 5.2 (a), (b), and (c). We can see that, first, when the number of packets increases, our algorithm performs better (i.e. end-to-end delay is smaller) if enough directions are available (here 3 IDPs are enough); second, the end-to-end delay of our algorithm is inversely proportional to the number of directions used. Note that when 7 IDPs are available, the end-to-end delay of our algorithm is about 65% of the HGR algorithm, 70% of that of the SPR and RandLLB algorithm.



(a) Comparsion with SPR



(b) Comparsion with HGR



(c) Comparsion with RandLLB

Fig. 5.2. Comparison of end-to-end delay

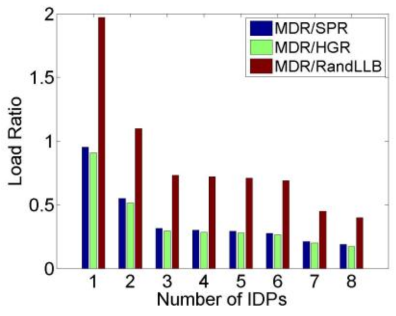


Figure 5.3. Comparison of load balance

E. Comparison of Load Balance

The comparison of load balance is showed in Fig. 5.3. From (5-4), we know that if R_L is less than 100%, it means our algorithm is better than other algorithms in respect of load balance. Fig. 5.3 shows that our algorithm has a smaller variance value, which means that packet flows distribute more smoothly than that of the SPR, HGR and RandLLB algorithm when enough directions are used. In addition, we can see that when more directions are used, the load is more balanced in our algorithm.

F. Comparison of Routing Table Size

Fig. 5.4 gives the comparison of routing table sizes for all algorithms. For our algorithm, when available directions increase, the size of the routing table also increases. When 8 IDPs are used, the routing table size of our algorithm is approximately twice of that of the SPR algorithm. When 4 IDPs are used, the routing table size of our algorithm is almost equal to that of SPR algorithm. Since HGR and RandLLB algorithm are both based on greedy routing without routing tables, we only compare with the SPR algorithm here. In exchange, these two algorithms pay the price for the worse performance on delivery ratio, delay ratio and load balance.

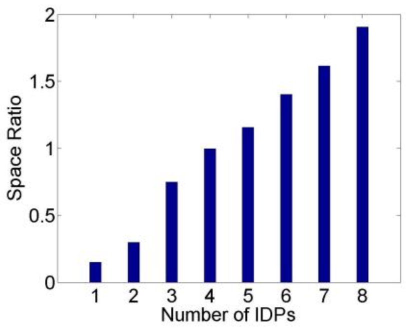


Figure 5.4. Comparison of routing table size

Remarks. As showed in the results, the tradeoff of our algorithm is to use more space to gain better performance on throughput, end-to-end delay and load balance. Our approach can obtain the remarkable improvement with acceptable space cost. For example, when we use seven IDPs, there is at least 20% improvement on throughput, end-to-end delay and load balance with the double of space cost. Besides, devices are equipped with more storage with the rapid growth of storage technologies. Therefore, our algorithm is feasible in practice.

VI. CONCLUSIONS

In this paper, we have discussed a new routing protocol, MDR. Compared to current geographical routing protocols, MDR uses direction as a metric for routing instead of using location as metric.

With the newly proposed direction metric, MDR provides a new solution on the issue of avoiding packets lost in local minimum that a geographical routing protocol should concern.

At the same time, MDR analyze its performance on the issues of throughput, end-to-end delay, load balance and network stability that previous geographical routing protocol often ignored. As proved, every source-destination pair can achieve a data rate $\Theta\left(1/\sqrt{NlogN}\right)$ simultaneously, while end-to-end delay is directly proportional to the network size. In simulation, we show that our method achieves good performance with acceptable space cost.

When the node density is sparse, the paths in different directions may have overlap parts. In the future, we will analyze the affect of the overlap parts of the paths to find the routing bottlenecks; then, to improve the MDR algorithm by discovering the alternative paths and deploying extra nodes along the bottleneck paths. Besides, we can further reduce the size of the routing table by applying the overlap analysis.

ACKNOWLEDGMENT

This work is supported in part by the National Basic Research(973) Program of China (Grant No.2011CB302502, 2011CB302803), the major national science and technology special projects(Grant No.2010ZX03004-003-03) and the NSF grant No. 0943479.

REFERENCES

[1] E. A. Lee, "Cyber physical systems: Design challenges," in *IEEE Proc. ISORC*, 2008.

[2] B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proc. MobiCom*, 2000.

[3] J. Duan, D. Li, W. Chen, and Z. liu, "3D geometric routing without loops and dead ends in wireless sensor networks," *Elsevier Journal of Ad Hoc Networks*, 13: 312-320, 2014.

[4] H. Frey and I. Stojmenovic, "On delivery guarantees of face and combined greedy-face routing algorithms in ad hoc and sensor networks," in *Proc. MobiCom*, 2006.

[5] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Worst-case optimal average-case efficient geometric ad-hoc routing," in *Proc. MobiHoc*, 2003.

[6] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," in *Proc. MobiCom*, 2003.

[7] F. Kuhn, R. Wattenhofer, and A. Zollinger, "An algorithmic approach to geographic routing in ad hoc and sensor networks," *IEEE/ACM Trans. on Networking*, vol. 16, 2008.

[8] S. Lam and C. Qian, "Geographic routing in d-dimensional spaces with guaranteed delivery and low stretch," in *Proc. SIGMETRICS*, 2011.

[9] Y. Li, Y. Yang, and X. Lu, "Routing metric designs for greedy, face and combined-greedy-face routing," in *Proc. Infocom*, 2009.

[10] S. Ruhrup, H. Kalosha, A. Nayak, and I. Stojmenovic, "Message-efficient beaconless georouting with guaranteed delivery in wireless sensor, ad hoc, and actuator networks," *ACM /IEEE Trans. on Networking*, vol. 18, 2010.

[11] S. Xia, X. Yin, H. Wu, M. Jin, and X. Gu, "Deterministic greedy routing with guaranteed delivery in 3D wireless sensor networks," in *Proc. ACM MobiHoc*, 2011.

[12] A. Cvetkovski and M. Crovella, "On the choice of a spanning tree for greedy embedding of network graphs," *Springer Journal of Networking Science*, vol. 3, pp. 2-12, 2013.

[13] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *Proc. SIGCOMM*, 2004.

[14] D. Eppstein and M. Goodrich, "Succinct greedy geometric routing using hyperbolic geometry," *IEEE Trans. on Computers*, vol. 99, 2010.

[15] R. Jiang, X. Ban, M. Goswami, W. Zeng, J. Gao, and X. Gu, "Exploration of path space using sensor network geometry," in *ACM/IEEE Proc. of IPSN*, 2011.

[16] Y. Jiang, W. Shi, X. Wang, and H. li, "A distributed routing for wireless sensor networks with mobile sink based on the greedy embedding," *Elsevier Journal of Ad Hoc Networks*, vol. 20, pp. 150-162, 2014.

[17] R. Kleinberg, "Geographic routing using hyperbolic space," in *Proc. INFOCOM*, 2007.

[18] R. Sarkar, X. Yin, J. Gao, F. Luo, and X. Gu, "Greedy routing with guaranteed delivery using ricci flows," in *Proc. ACM/IEEE IPSN*, 2009.

[19] C. Qian and S. Lam, "Greedy distance vector routing," in *Proc. IEEE ICDCS*, 2011.

[20] S. Douglas, J. De Couto, and R. Morris, "Location proxies and intermediate node forwarding for practical geographic forwarding," *MIT-LCS-TR-824, MIT Laboratory for Computer Science*, 2001.

[21] S. Subramanian, S. Shakkottai, and P. Gupta, "On optimal geographic routing for networks with holes and non-uniform traffic," in *Proc. IEEE Infocom*, 2007.

[22] S. Subramanian, S. Shakkottai, and P. Gupta, "Optimal geographic routing for wireless networks with near-arbitrary holes and traffic," in *Proc. IEEE Infocom*, 2008.

[23] L. Valiant and G. Brebner, "Universal schemes for parallel communication," in *Proc. 13th annual ACM symposium on Theory of Computing*, New York, NY, USA, 1981, pp. 263–277.

[24] Y. Ko and N. Vaidya, "Location-aided routing in mobile ad hoc networks," in *Proc. ACM MobiHoc*, 1998.

[25] Y. Wang, J. Gao, and J. Mitchell, "Boundary recognition in sensor networks by topological methods," in *Proc ACM MobiCom*, 2006

[26] D. Dong, Y. Liu, and X. Liao, "Fine-grained boundary recognition in wireless ad hoc and sensor networks by topological methods," in *Proc. ACM MobiHoc*, 2009.

[27] H. Zhou, H. Wu, and M. Jin, "A robust boundary detection algorithm based on connectivity only for 3D wireless sensor networks," in *Proc. IEEE Infocom*, 2012.

[28] B. Hajek and G. Sasaki, "Link scheduling in polynomial time," *IEEE Trans. on Information Theory*, vol. 34, no. 5, pp. 910–917, 1988.

[29] IEEE Std. 802.11: Wireless LAN Media Access Control (MAC) and Physical Layer (PHY) Specifications, 1999.

[30] P. Gupta and P. R. Kumar, "Critical power for asymptotic connectivity in wireless networks," *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W. H. Fleming*, pp. 547–566, 1998.

[31] R. Motwani and P. Raghavan, *Randomized Algorithms. Cambridge*, U.K.: Cambridge University Press, 1995.

[32] D. Bertsekas and R. Gallager, *Data Networks*, U.K. Prentice Hall, 1987, ch.4.

[33] L. Tassiulas and A. Ephremide, "Stability properties of constrained queuing systems and scheduling for maximum throughput in multihop radio networks," *IEEE Trans. on Automatic Control*, pp. 1936–1949, 1992.

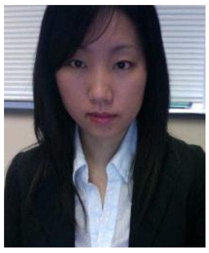
[34] X. Lin and N. B. Shroff, "Joint rate control and scheduling in multihop wireless networks," in *Proc. IEEE CDC*, 2004.

[35] X. Lin and N. B. Shroff, "The impact of imperfect scheduling on crosslayer control in multihop wireless networks," in *Proc. IEEE Infocom*, 2005.

[36] S. Sastry, *Nonlinear Systems: Analysis, Stability and Control*, U.K.: Addison Wesley, 1996.

Zimu Yuan received the B.S degree in computer science from Nanjing University in 2009. He is currently pursuing his studies toward a Ph.D. degree in the Institute of Computing Technology, Chinese Academy of Sciences (ICT, CAS). His research interests include wireless communication, localization and geographic routing.

Wei Li is an Associate Professor at the Institute of Computing Technology, Chinese Academy of Sciences (ICT, CAS). His research interests include distributed system, parallel computing architecture, and big data analytic. He has published more than 40 papers in conferences and journals related to these research areas. Dr. Wei Li received his Ph.D. degree from Institute of Computing Technology, Chinese Academy of Sciences (ICT, CAS) in 2008. He is a member of the IEEE and the IEEE Computer Society.



Shuhui Yang is currently an assistant professor with Department of Math, Computer Science, and Statistics of Purdue University Calumet. She received a Ph.D. degree from Florida Atlantic University, USA in 2007. She has published more than 30 papers in various peer-reviewed journals and conference proceedings. Dr. Yang was guest editor of the EURASIP (European Association for Signal Processing) Journal on Wireless Communications and Networking, Special issue on Wireless Network Security, 2009 and member of Guest Editorial Board for International Journal of Communication Networks and Distributed Systems (IJCNDS), 2008. She served as Program co-chair for the 6th WiSARN 2012-Fall. She also served as Program Committee member for IPCCC, WSNS, ICDCS, WiSP, and SN among many others. She served as NSF panelist for 2008. She won the best paper award of the 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS). She is a member of the IEEE.