Copyright © 2012 KSII

Neighbor Gradient-based Multicast Routing for Service-Oriented Applications

Hui Wang, Jianbiao Mao, Tao Li, Zhigang Sun,
Zhenghu Gong and Gaofeng Lv
School of Computer, National University of Defense Technology
Changsha, Hunan 410073 – P.R.China
[e-mail: wanghuinudt@gmail.com]
*Corresponding author: Hui Wang

Received April 9, 2012; revised May 28, 2012; revised July 14, 2012; revised August 19, 2012; accepted September 11, 2012;published September 26, 2012

Abstract

With the prevalence of diverse services-oriented applications, such as IPTV systems and on-line games, the current underlying communication networks face more and more challenges on the aspects of flexibility and adaptability. Therefore, an effective and efficient multicast routing mechanism, which can fulfill different requirements of different personalized services, is critical and significant. In this paper, we first define the neighbor gradient, which is calculated based on the weighted sum of attributes such as residual link capacity, normalized hop count, etc. Then two distributed multicast routing algorithms which are neighbor Gradient-based Multicast Routing for Static multicast membership (GMR-S) and neighbor Gradient-based Multicast Routing for Dynamic multicast membership (GMR-D), are proposed. GMR-S is suitable for static membership situation, while GMR-D can be used for the dynamic membership network environment. Experimental results demonstrate the effectiveness and efficiency of our proposed methods.

Keywords: Multicast, service-oriented application, distributed, neighbor gradient

A preliminary version of this paper appeared in IEEE ICMT 2011, July 26-28, Hangzhou, China. This version includes a concrete analysis and supporting implementation results on neighbor gradient based multicast routing. This research was funded by a grant from the Major State Basic Research Development Programs of China (973 Programs) (No.2009CB320503) and National Natural Science Foundation of China (No.61003301).
http://dx.doi.org/10.3837/tiis.2012.09.015

1. Introduction

With the development of Internet, more and more service-oriented applications have been emerging in the recent years. Such applications include IP Television (IPTV), network games, live video-teleconferences, stock exchange, etc [1]. These diverse applications involve a source sending packets to a selected set of destinations. Therefore, IP multicast technologies, which support the underlying networks to save bandwidth and network resource, are critical and significant.

Meanwhile, the diversity of service-oriented applications also brings new challenges to the multicast routing. The requirements of multicast routing in service-oriented applications can be summarized as follows:

Adaptability of multicast routing to different traffic requirements. Different applications and services have different QoS requirements. For example, live video-teleconference requires high occupied bandwidth and stringent end-to-end delay, while transferring stock information and data broadcast require less bandwidth and lower delay sensitivity. Meanwhile, some applications may only ask for the lowest requirements, such as delay-tolerant data delivery, which only requires the delay bound in a reasonable range. Since different types of applications have different flow characteristics, it is significant to design an adaptive multicast routing policy to fulfill different requirements of diverse applications.

Avoiding congestion and improving load balance. Too many application streams centralizing at some certain hot spots over the networks will lead to less multicast requests being satisfied. What is worse, this makes a small part of multicast sessions over-occupy the large part of resources at some nodes and some links, which badly influences the performance of upcoming multicast requests. Hence, an adaptive multicast routing, which can keep fairness among multiple multicast groups, is quite challenging yet much desired.

Supporting dynamic behaviors of users. In real networks, the members join or leave the specified groups randomly and frequently. This makes the routing policy must be adapted to dynamic behaviors and time-varying activities in wide-area networks. Reconstructing the existing multicast routing tree should be considered when multicast membership has been dynamically changed.

Distributed routing for large scale groups and members. The sizes of the multicast group for current service-oriented applications increase continuously. The latest forecast indicated the global IPTV subscribers would come to 111.5 million in 2014, with a compound annual growth rate of 26% [2]. Each channel of IPTV system can be a potential multicast group. With the increasing numbers of groups and members, the appropriate multicast routing should be computed in time at each node based on its latest updated information but not the static and outdated information.

Though there are already a lot of researches produced for multicast routing problem, designing an adaptive and flexible multicast routing for diverse service-oriented applications is still extremely challenging yet much desired. That is because, on the one hand, the existing static multicast routing algorithms [3][4][5] just consider the static group situation, in which membership remains unchanged throughout the group's lifetime. Factually, in most of real situations, memberships are changed dynamically. On the other hand, some dynamic multicast routing algorithms [6][7] have been proposed for dynamic situation, wherein the membership of the multicast group changes with time. However, such works either focus on a single factor

[3][4] or a specific application for multicast routing. Moreover, these routing algorithms are unable to fulfill the requirements of different types of service-oriented applications in the current Internet architecture [8].

For example, in the traditional IP multicast service model, the most used multicast routing protocol is PIM-SSM [9], which is based on the Shortest Path Tree (SPT) and unicast routing protocols. So it is rigid and ossified for employing only one multicast routing scheme to meet all types of traffic requirements. Hence, we need a generic flexible multicast routing to adapt to different policies. It should use specific hybrid metrics which can accommodate a variety of service-oriented applications. And it can ensure that packets avoid congested areas but do not traverse the network using random walks. Therefore, in order to sufficiently provide personalized services to end users, a more flexible multicast routing is critical and significant.

In this paper, we first define a neighbor gradient by simultaneously considering several key factors, which influence the QoS of the multicast routing. Based on the neighbor gradient definition, two distributed neighbor gradient based multicast routing algorithms, which are designed for static multicast membership and dynamic multicast membership respectively, are proposed. The contributions of this paper are summarized as follows:

- In order to find the effective and efficient multicast routing, we define a novel neighbor gradient, in which several important multicast routing factors such as the scale of the multicast tree, link load of the network and hop count of the routing path, are considered simultaneously.
- Two distributed neighbor gradient based multicast routing algorithms, which are GMR-S and GMR-D are proposed respectively. GMR-S is designed for static multicast membership situation, while GMR-D is fit for dynamic multicast membership situation.
- We also present the extensive evaluation analysis to show that our proposed distributed multicast algorithm for static multicast membership can achieve much better performance compared with two classical multicast routing algorithms. Further experiments demonstrate that our dynamic multicast algorithm is effectively adaptive to dynamic membership situation as well.

The remaining part of this paper is organized as follows: Section 2 introduces the related work. In section 3, a novel neighbor gradient is defined, and two adapted multicast routing algorithms are described and analyzed in detail. Simulation results are discussed in section 4. Finally, conclusions are given in section 5.

2. Related Work

The related work is given from two aspects. We first introduce the classical multicast service models. Then, we explain the related applications of the potential field based routing since our proposed algorithms are based on the field theory and steepest gradient search from physics area.

Protocol Independent Multicast (PIM) is an inter-domain protocol which supports both shared and source-specific distribution trees [4]. However, QoS is not taken into account in route selection in PIM protocols. PIM-Source Specific Multicast (PIM-SSM) [9] is the most widely used multicast service model in IPTV systems. In PIM-SSM, Shortest Path Tree (SPT) is employed from its source to group destinations no matter what type of the traffic is. Most multicast routing problems can be formulated as a Steiner Tree (ST) or constrained ST problem [3] and there are so many centralized or distributed algorithms to solve them, such as

KMB [5], Jia [7], QDMR [10], WAVE [11], etc. Those QoS-oblivious [4][12] or QoS-sensitive [10][13] algorithms just try to heuristically construct a low-cost tree for one type of application, and some may subject to a given upper bound on end-to-end delay. Hence they are not designed specially for diverse service-oriented applications. Yang et al. [14] proposed a service-centric multicast architecture and a Service Centric Multicast Protocol (SCMP) for providing efficient and flexible multicast service over the Internet. SCMP adopted the DCDM algorithm [15] for constructing the multicast tree. DCDM is a centralized algorithm and only considers link delay and link cost. [16] proposed a scalable and adaptive protocol for multicast communications. The protocol is scalable in terms of both the group number and group size. However, its main object only focuses on the scalability problem, but not the adaptability problem in multicast routing. In order to support QoS IPTV service, Wen et al. proposed a hybrid tree based explicit routed multicast (HT-ERM) approach for IPTV service [17], which combines the advantages of Rendezvous Point based shared tree (RPT) and specific source based SPT in traditional IP multicast protocol. However, HT-ERM is only suitable for IPTV application, but not universal for other service-oriented applications. Cho et al. presented a MAD (Multicast with Adaptive Dual-state) architecture to provide efficient multicast service at massive scale [8]. MAD has two transmission modes: it uses IP multicast-style dissemination tree to deliver messages for active groups, and membership tree based forwarding to deliver messages for inactive groups. MAD can mitigate IP multicast and overlay multicast when there are a large number of groups with long-live data traffic, but cannot fundamentally solve the adaptability issue in multicast routing.

Recently, potential-based routing has been proposed for various types of applications, including routing in load balancing of the Internet, data collection, node placement in sensor networks, and service discovery in MANETs, etc [18]. [19] presented a Potential-based Multicast Tree Algorithm (PMTA) to enhance the multicast tree construction in presence of connectivity restricted hosts. A potential-based routing was proposed in [20] which can be adapted to provide dynamic, traffic-aware routing by designing a traffic-based potential. Balasubramaniam et al. [21][22] proposed a new resource management scheme which incorporated a new gradient based routing mechanism-- PGBR, to deliver IPTV content over an IP network. However, most of potential-based multicast routing algorithms are used for unicast routing or provided for specific situations such as wireless networks. And they can not be applied for multicast routing situations directly.

As discussed above, there are a number of challenges to address in multicast routing for service-oriented applications. Our work differs from the existing works as follows. Firstly, we propose the neighbor gradient definition, which can be used for static multicast routing as well as dynamic multicast routing. Secondly, since our multicast routing algorithms consider different critical factors simultaneously, they can be applied for various service-oriented applications. Thirdly, as we will see in the experimental part, our algorithms have a much better performance compared with existing solutions.

3. Neighbor Gradient-based Multicast Routing

In this section, we propose two multicast routing algorithms, which are neighbor Gradient-based Multicast Routing for Static group membership (GMR-S) and neighbor Gradient-based Multicast Routing for Dynamic group membership (GMR-D). GMR-S is suitable for static membership situation, while GMR-D can be used for the dynamic membership network environment. Both GMR-S and GMR-D are distributed multicast routing algorithms, which can be executed in different nodes of the flow network.

The idea is inspired by the field theory and steepest gradient search from the physical area. Our method focuses on selecting the steepest neighbor gradient path from source to each destination, which means packets are forwarded towards the direction of maximum positive force in the network potential field. More specifically, the forwarding progress can meet various types of demands for service-oriented applications effectively, and minimize the whole network potential energy reduction.

Our proposed solution has the following functionalities:

1) Maximize the reuse gain of the existing tree structure when discovering the routing path from the source to a new destination.

2) Control the overload of those hot links which may be shared by the paths from the source to many destinations.

3) Find a shorter path to meet the delay or hop requirements for each destination.

Before explaining our algorithms in detail, some important definitions are given.

3.1 Definitions

Definition 1. Flow Network. A flow network is described as a directed graph $F=(V,E,S,D)$, where V is the node set, and E is the set of directed edges, $S \subseteq V$ is source set and $D \subseteq \{V-S\}$ is destination set. The information flow streams from a source node $s \in S$ to all of the destinations $d \in D$.

Flow networks widely exist in the real-world networks. For example, Internet is one of the most universal flow networks. Besides, from Definition 1, we know that it is possible to have multiple sources and destinations in a flow network. For the convenience of discussion, we only consider the situation of one source and multiple destinations in this paper. Specially, we use $F=(V,E,s,D)$ to stand for the flow network when there is the single source. In the rest of this paper, we only consider the situation of the flow network with a single source. However, it is easy to extend our proposed methods to multiple sources and multiple destinations situation.

Definition 2. Multicast Tree. Given a flow network $F=(V,E,s,D)$, a multicast tree $T=(V_T,E_T,s,D)$ ($V_T \subseteq V$, $E_T \subseteq E$) is a special tree, which is rooted at s and reaches each destination $d \in D$.

Different from the gradient definitions proposed in [21] and [22] for unicast routing, we consider the ability of sharing path, residual link capacity and normalized hop count simultaneously when constructing the multicast tree. That is because there are critical factors for the high quality of multicast routing in most of the service-oriented applications. The neighbor gradient is defined as follows [23]:

Definition 3. Neighbor Gradient. Given a flow network $F=(V,E,s,D)$, for the source node s and the specified destination node $d \in D$, the neighbor gradient between any pair of neighbor nodes $u,v \in V$ for destination d at time t is defined as follows:

$$G_{u\to v,(s,d)}(t)=\alpha\varphi_v(t)+\beta l_{u\to v}(t)+\gamma h_{v,d} \tag{1}$$

where $\varphi_v(t)$, which is further described in Equation (2), measures the ability of neighbor node v for forwarding the upcoming packets from its upstream node u in multicast tree T, share the existing paths in multicast tree $l_{u\to v}(t)$ represents the residual capacity of the link $e(u,v)$ at time t, $h_{v,d}$ denotes the normalized total hop counts from node v to destination d,

and α, β and γ are the weight values for each respective term in Equation (1) ($0 \le \alpha, \beta, \gamma \le 1$, $\alpha + \beta + \gamma = 1$).

$$\varphi_v(t) = \begin{cases} 0, \text{ if } v \notin T \text{ at time } t \\ 1/i, \text{ if } v \in T \text{ at time } t \text{ and the current node } d \text{ is the i}^{\text{th}} \text{ nearest destination from } s \end{cases} \quad (2)$$

As we mentioned, $\varphi_v(t)$ measures the ability of the neighbor node v for forwarding the upcoming packets from its upstream node u in multicast tree T. This is accomplished by assigning an appropriate value to $\varphi_v(t)$ through considering whether the neighbor node v has already been in the multicast tree T or not. In detail, on the one hand, if node v is not included in the multicast tree, i.e., $v \notin T$, we set $\varphi_v(t) = 0$. That is because when the current neighbor node v is not contained in the current multicast tree, the routing path of destination d can not share more existing routing paths, which are already in the multicast tree T, by selecting current neighbor node v as the next hop. On the other hand, if the current neighbor node v is on the multicast tree T, i.e., $v \in T$, it is apt to be selected as the next hop when discovering the routing path for destination d. Moreover, the value of $\varphi_v(t)$ should consider the distance of between source s and the current destination d. In other words, the importance of sharing the existing paths in the multicast tree should be weakened when the current destination is far from the source. That is because the destinations, which have longer distance from the source node, will have larger delay. In this case, if we still focus too much on sharing the existing routing path of the multicast tree T, it will make such destinations, which have longer distance from the source, have even longer routing path and even larger delay. Hence, we should distinguish the destinations, which have shorter routing paths, from the other destinations, which have longer routing paths, when consider the value of $\varphi_v(t)$. Therefore, we set $\varphi_v(t)$ to $1/i$ but not to 1 (See Equation (2)), where i is the distance order from source s to the current destination d. The experiment part demonstrates that this function works very well on balancing the length of routing paths and the width of the multicast tree.

The second term $l_{u \to v}(t)$ in Equation (1) is the residual link capacity of link $e(u,v)$ at time t. The residual link capacity will be updated if it is beyond a certain threshold during the routing process. For instance, if the residual link capacity exceeds the threshold, the new residual link capacity $l_{u \to v}(t+1)$ will be recomputed and sent to the related nodes. Then the neighbor gradient will be calculated again based on the new residual link capacity.

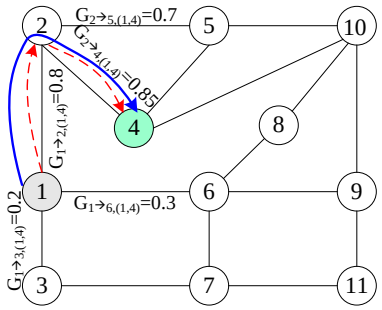
The last term h_{v,d_i} in Equation (1) is the normalized hop count between current neighbor node v and the destination d. The normalized hop count, which is defined in Equation (3), indicates the relative distance between the neighbor node v and the specified destination d. It can be calculated at the network initialization phase and will not change again unless the network topology is restructured.

$$h_{v,d} = 1/H(v,d) \text{ (routing path: } s \to \ldots \to v \to \ldots \to d) \quad (3)$$

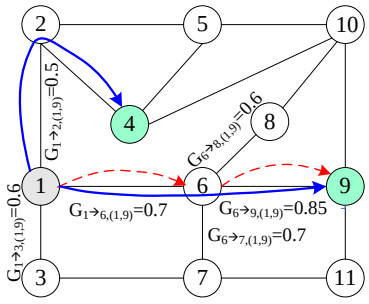
Generally, for any node $u \in V$ and one of its neighbor node $v \in V$ in flow network $F = (V, E, s, D)$, the higher ability of node v for making the current routing path, which starts from source s and ends at destination d, share more existing paths in current multicast tree T, the larger residual capacity in link $e(u,v)$, and the shorter normalized hop counts from current neighbor node v to the specified destination d will lead to the larger value of $G_{u \to v,(s,d)}(t)$. This indicates that, from the point of node u, the current neighbor node v should be chosen as the next hop with a higher priority than other neighbor nodes of node w.

The calculation of neighbor gradient value of each link for a specified destination will be accomplished during the routing process. In order to further explain the definition of neighbor gradient, an illustrative example for the neighbor gradient search process is given in **Fig. 1**. In this example, given a flow network F with 10 nodes, the source node $s=1$ and the destination set $D=\{4,9,10,11\}$, we illustrate the neighbor gradient search process for destination nodes 4, 9, 10, 11 in **Fig. 1-(a)**, **Fig. 1-(b)**, **Fig. 1-(c)** and **Fig. 1-(d)** respectively. The directed dash red lines in each subfigure illustrate the process of discovering routing path, and the directed solid blue lines denote the discovered routing paths, which are a part of multicast tree. The discovery routing path for each destination starts from the source node. The destinations are added to the multicast tree orderly according to the distance between the source and the specified destination. As we discussed before, the destinations, which have a shorter distance from the source, will be added to the multicast tree prior to destinations, which have a longer distance from the source.

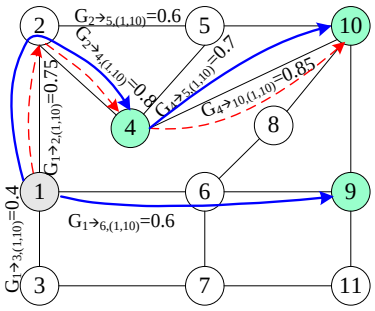
Hence, among all of the destinations, i.e., nodes 4, 9, 10, 11, the closest destination node 4 will be added to the multicast tree first. The process of discovering the routing path for destination 4 is illustrated in **Fig. 1-(a)**. Aiming at destination 4, we start at the source node 1. Initially, we choose the neighbor node 2, which has the highest neighbor gradient among all of the neighbor nodes of source node 1, as the next hop of source 1. Subsequently, we check the neighbors of node 2 and choose the neighbor node 4, which has the highest neighbor gradient among all of the neighbors of node 2, as the next hop of node 2. Since the current node 4 is already our destination, it means that we have already successfully discovered the routing path $1\rightarrow 2\rightarrow 4$ for destination 4. The routing path discovery processes for destinations 9, 10 and 11 are very similar to the process of discovering the routing path for destination 4, and illustrated in **Fig. 1-(b)**, **Fig. 1-(c)** and **Fig. 1-(d)** respectively. The only difference between the processes of discovering routing paths for destinations 4, 9 and the processes of discovering routing paths for destinations 10, 11 is that, the routing paths of destinations 10 and 11 can effectively share the existing paths of the multicast tree, while the routing paths of destinations 4 and 9 have no existing paths to be reused during the discovering routing paths processes. After the routing paths for all of the destinations being discovered, the final multicast tree, which is exhibited in directed solid blue line in **Fig. 1-(d)**, is created.



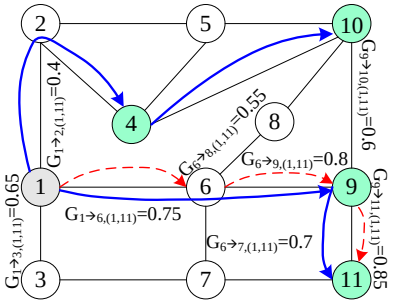
(a) Routing path discovery for destination 4



(b) Routing path discovery for destination 9



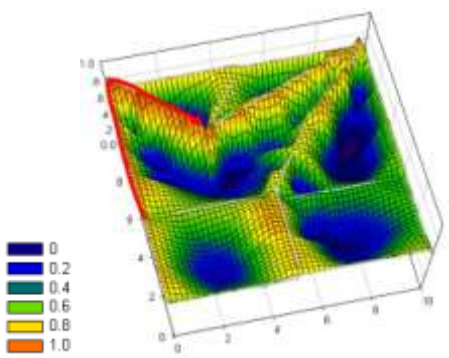
(c) Routing path discovery for destination 10



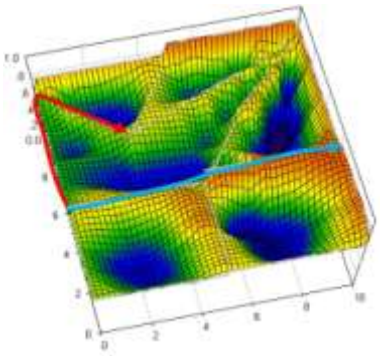
(d) Routing path discovery for destination 11

Fig. 1. The neighbor gradient search process for different destinations

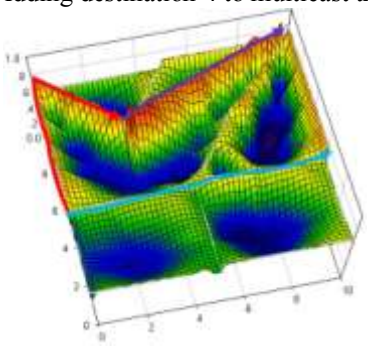
In **Fig. 2**, we depict a 3-dimensional illustration of the neighbor gradient search process starting from source 1 to destinations 4, 9, 10 and 11. These figures show how different paths are taken for corresponding destinations during the neighbor gradient search process.



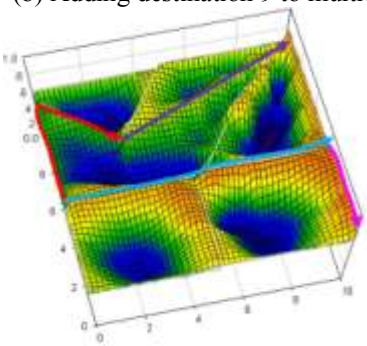
(a) Adding destination 4 to multicast tree



(b) Adding destination 9 to multicast tree



(c) Adding destination 10 to multicast tree



(d) Adding destination 11 to multicast tree

Fig. 2. Potential field illustration for neighbor gradient search

3.2 Static Multicast Membership

In static multicast membership situation, the multicast destinations are not changed during the whole multicast session. The neighbor Gradient-based Multicast Routing for Static group memberships (GMR-S) is given in **Fig. 4**. Since GMR-S is based on message transfer, we introduce two important types of messages used in GMR-S before discussing our algorithm further.

Discovery message (*Dis_msg*): Discovery messages are used to discovery the routing path for each destination. The important routing information, such as the number of current hop counts and traversed nodes, are recorded in *Dis_msg* . Source node s initializes and sends the discovery message to its neighbor, which has the highest neighbor gradient among all of the neighbors. Then the discovery message will be forwarded towards the direction of the steepest neighbor gradient, which is described in Section (4.1), until the specified destination has been found or the maximum hop count has been reached.

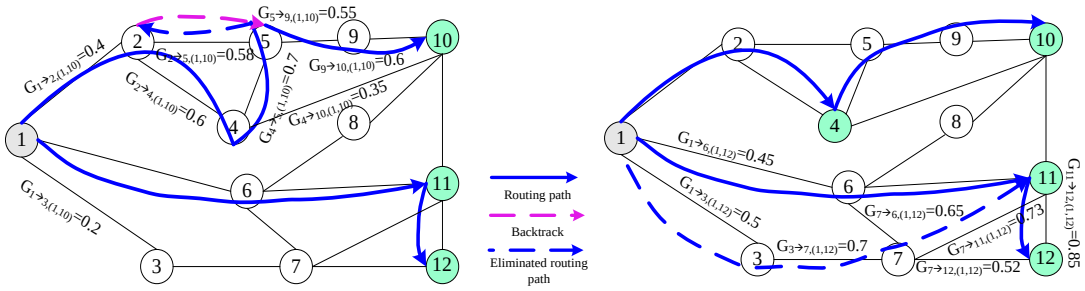
Feedback message (*Fdb_msg*): Feedback messages are used to notify the related nodes to generate the correct multicast tree. After the routing path of the specified destination has been discovered, the destination transforms the discovery message to feedback message. The feedback message will be forwarded to source s along the discovered routing path, which is recorded in *Fdb_msg* . Consequently, the nodes, which are in the routing path, can grow the multicast tree further.

Even though discovery message and feedback message have different functions, these two types of messages share the similar data structures. The main components of data structure include five fields: (1) Message type (*MT*). *MT* field indicates the type of this message belongs to *Dis_msg* or *Fdb_msg* . (2) Source and Destination (*SD*). *SD* field records the source and the destination of current routing process. (3) Visited Node List (*VNL*). *VNL* field records the nodes, which have been visited during the process of discovering the current routing path. (4) Current Path Length (*CPL*). *CPL* field records the length of the current discovered routing path. (5) Maximum Path Length (*MPL*). *MPL* records the threshold of the maximum hop count that the message is allowed.

As described in **Fig. 4**, the process of GMR-S can be divided into two stages. The main task of the first stage is initialization, which includes steps (1) ~ (7) in **Fig. 4**. In the network initialization process, Floyd-Warshall algorithm [24] is used to compute the distance between the current node u and the specified destination d . Then we compute the $h_{v,d}$ according to Equation (3). The time complexity of the network initialization process is $O(|V|^3)$. Since this process can be finished off-line, the computation cost of the initialization process does not affect the efficiency of GMR-S. After steps (2) ~ (7), the source node s has already kept all of the static destinations into its destination list DL according to the ascending order of the distance between source s and the specified destination d .

The main task of the second stage is to discover the routing paths from the source s to all destinations. This stage includes steps (8) ~ (38) in **Fig. 4**. Firstly, as shown in steps (9) ~ (11), source s fetches the closest destination from the current destination list *DL* and sends the discovery message *Dis_msg* to its neighbor, which has the highest neighbor gradient among all of its neighbors. Then, the *Dis_msg* message will be forwarded further from node to node towards the direction of steepest neighbor gradient. The forwarding process is demonstrated from step (13) to step (15). When a node u receives the *Dis_msg* message from its neighbor, u will select the node with the highest neighbor gradient to forward the *Dis_msg* message. During the forwarding process, all of the visited nodes will be recorded in *Dis_msg.VNL* as the possible routing path for the current destination, and the field *Dis_msg.CPL* will also be updated by the forwarding node. As described above, the message transmission process will perform hop-by-hop until the *Dis_msg* message has reached the specified destination d . In order to discover an effective routing path, two types of loops need to be eliminated carefully.

The first type of loop is *current routing path loop* (see **Fig. 3-(a)**). This type of loop is generated when the nodes, which are already visited are stored in *Dis_msg.VNL*, are visited again. As described in steps (16) ~ (20), to eliminate this type of loop, we only need to backtrack the current discovery message *Dis_msg* to the previous hop and choose the second highest gradient neighbor node as the next hop. Then we update the related fields such as *VNL* and *CPL* in *Dis_msg*. **Fig. 3-(a)** illustrates an example of the current routing path loop and the mechanism about how to eliminate this type of loop. Assume discovery message *Dis_msg* is forwarded to node 5 at time t. Therefore, the current routing path stored in *Dis_msg.VNL* is $1\rightarrow 2\rightarrow 4\rightarrow 5$. Then, node 5 forwards the discovery message *Dis_msg* to node 2, because node 2 has the highest gradient among all of the next hop neighbors of node 5. Since node 2 has already been recorded in the current routing path, the current routing path loop is formed. In this case, the discovery message *Dis_msg* trackbacks to node 5. Subsequently, node 5 selects the second highest gradient neighbor to forward the discovery message *Dis_msg*. At this moment, the new routing path is $1\rightarrow 2\rightarrow 4\rightarrow 5\rightarrow 9$ instead of $1\rightarrow 2\rightarrow 4\rightarrow 5\rightarrow 2$. Hence, the final routing path for destination 10 will be $1\rightarrow 2\rightarrow 4\rightarrow 5\rightarrow 9\rightarrow 10$.



(a) Eliminating the first type of loop (b) Eliminating the second type of loop

Fig. 3. The types of loops and elimination techniques

The second type of loop is *existing routing path loop* (see **Fig. 3-(b)**). This type of loop is formed when the nodes, which have already existed in the current multicast tree *T*, are traversed again. Assume the current node *v*, which has already been in the multicast tree, is visited again. In order to eliminate the second type of loop, as described in steps (21) ~ (24), node *v* only need to clean the discovered routing path in *Dis_msg.VNL*. Then node *v* continues to forward discovery message *Dis_msg.VNL* to its next hop. Since our algorithm is de-centralized, the multicast tree is preserved in each node separately. In other words, this operation will not miss the feedback message coming from the destination. Moreover, this operation can further reduce the total links of the multicast tree and improve the routing efficiency. As we will see in the experiment part, for the excellent definition of the neighbor gradient and the specific techniques used to eliminate different types of loops, the final multicast tree improves the routing efficiency significantly. An example about how to break the second type of loop is illustrated in **Fig. 3-(b)**. Suppose the discovery message *Dis_msg* has arrived at node 7 when discovering routing path for the destination node 12. Obviously, the current routing path is $1\rightarrow 3\rightarrow 7$. Then node 7 will continue to forward the discovery message *Dis_msg* to node 11 since node 11 has the highest neighbor gradient. At this moment, we get the second type of loop, because node 11 has already existed in the discovered

multicast tree T. In this situation, node 11 clears the original routing path $1 \rightarrow 3 \rightarrow 7 \rightarrow 11$ and forwards discovery message *Dis_msg* to the next hop. At last, we will get the routing path $1 \rightarrow 6 \rightarrow 11 \rightarrow 12$ for destination 12.

Besides, in order to further improve the quality of the routing path for each destination, at steps (25) ~ (29), we set a minimal neighbor gradient threshold G_T to filter the nodes whose neighbor gradient is less than the threshold. Steps (30) ~ (32) show that the discovery process also has a path length limitation. In other words, once the length of the current routing path exceeds the maximum path length threshold (MPL), the route discovery will terminate. Finally, as described from steps (33) ~ (36), if the discovery message *Dis_msg* reaches the specified destination within *Dis_msg.MPL* hop counts, the destination node transfers the discovery message *Dis_msg* into a feedback message *Fdb_msg* and sends it back to the source s along the just discovered routing path. Otherwise, the routing path discovery for this destination fails. The above neighbor gradient search process for a specified destination repeated many times, until all of the destinations are added into the multicast tree.

The neighbor gradient based multicast routing pseudo code is shown in **Fig. 4**.

Algorithm 1 Neighbor Gradient-based Multicast Routing for Static Multicast Membership

Input: Flow network $F = (V, E, s, D)$;
Maximum path length threshold (*MPL)*;
Minimum neighbor gradient threshold G_T

Output: Multicast tree T

1: Network_Initialization ($F(V, E, s, D)$);
2: $T \leftarrow s$;
3: **for** source s
4: **for** each $d \in D$
5: Store d in DL according to ascending order of the distance between s and d ;
6: **end for**
7: **end for**
8: **while** $DL \neq \varnothing$
9: Fetch d from the head of DL ;
10: Initialize *Dis_msg* ;
11: Select the steepest $G_{s \rightarrow u}$ to forward *Dis_msg* ;
12: **for** node $u \in V$
13: $u \in V$ receives *Dis_msg* ;
14: u selects the steepest $G_{u \rightarrow v}$ ($v \in V$ is the neighbor node of u) to forward *Dis_msg* ;
15: u updates fields *VNL* and *CPL* in *Dis_msg* ;
16: **if** v is already in *VNL* **then**
17: v backtracks one hop to node u ;
18: u forward *Dis_msg* to the second steepest $G_{u \rightarrow w}$ ($w \in V$ is the neighbor node of u);
19: u updates fields *VNL* and *CPL* in *Dis_msg* ;
20: **end if**
21: **if** v is already in T **then**
22: Clear field *VNL* in *Dis_msg* ;

```
23:        Update VNL with the routing path from s to v in T;
24:      end if
25:      if G_{u→v} < G_T
26:        Dis_msg trackbacks one hop to node x ( x ∈ V is the latest node recorded
in VNL );
27:        x forward Dis_msg to the second steepest G_{x→y} ( y ∈ V is the neighbor
node of x );
28:        x updates fields VNL and CPL in Dis_msg;
29:      end if
30:      if Dis_msg.CPL > Dis_msg.MPL then
31:        Route discovery fails;
32:      end if
33:      if Dis_msg successful arrives at d then
34:        Turn Dis_msg as Fdb_msg and send it back to s along the
           discovered routing path in VNL;
35:        T ← T ∪ VNL;
36:      end if
37:    end for
38: end while
39: return T
```

Fig. 4. The de-centralized pseudo code for GMR-S

3.3 Dynamic Multicast Membership

In many real networks, a multicast group may allow new members to join, or old members to leave freely. In this situation, the multicast tree should be reconstructed dynamically according to the changes of the multicast membership during the whole session time. In this section, we introduce our neighbor Gradient-based Multicast Routing for Dynamic group membership (GMR-D) in detail. In order to explain clearly, given a flow network $F=(V,E,s,D)$ and its corresponding multicast tree T, we distinguish 4 different types of nodes in the network as follows:

Leaf node: For $\forall v \in V_T$, if v has no child, v is called a leaf node.

Branch node: For $\forall v \in V_T$, if v has more than two children, v is called a branch node.

Steiner node: For $\forall v \in V_T$, if $v \notin D$, where D is the destination set, v is called a steiner node.

Non-tree node: For $\forall v \in V$, if $v \notin V_T$, v is called a non-tree node.

When a node wants to join the multicast group, it sends a request message to the source. On the one hand, if the node is a steiner node, it can join the multicast tree directly since it has already been in the multicast tree. Subsequently, the new destination sends a feedback message to the upstream nodes, which will finally arrive at the source node. All the nodes, which the feedback message passes through, will be notified that the new destination has been added to the multicast tree. In this case, the new joined node will change its tag, and mark as a destination node. On the other hand, if a non-tree node wants to join the multicast group, it first sends a request message to the source. Then, the source sends a discovery message aiming at this non-tree node based on the highest neighbor gradient principle. This joining process will be similar with GMR-S for adding a new destination to the multicast tree. As described in

Definition 3, the nodes, which have already been in the multicast tree, are easier to get the higher neighbor gradient. Therefore, the new routing path will have a higher chance to share more links in the multicast tree. The process of a node joining a multicast group is described in **Fig. 5**.

Algorithm 2 New Destination Joining Process

Input: Flow network $F=(V,E,s,D)$; Multicast tree T

Output: New multicast tree T_{new}

1: **for** the joining node v
2: **if** $v \in V_T \cap v \notin D$ **then**
3: Mark v as a destination node;
4: Send the feedback message Fdb_msg along the upstream nodes to source;
5: $T_{new} \leftarrow T$;
6: **end if**
7: **if** $v \notin V_T$ **then**
8: Call GMR-S for adding the new destination v into the multicast tree T and generate the new multicast tree T_{new};
9: **end if**
10: **end for**
11: **return** T_{new}

Fig. 5. New destination joining process for GMR-D

When a destination wants to leave the multicast group, GMR-D takes different actions for the different types of nodes. If the destination is a leaf node, it first sends a leave message to the upstream nodes in the multicast tree. Then, this leave message is forwarded until it reaches a branch node of the multicast tree. All the nodes, which have received the leave message, release the connection for this destination. Consequently, the related nodes and links are deleted from the multicast tree. If the leaving destination is not a leaf node, it only needs to mark itself as a steiner node and continue to perform the forwarding function. The process of a node leaving the multicast group is described in **Fig. 6**.

Algorithm 3 Destination Leaving Process

Input: Flow network $F=(V,E,s,D)$; Multicast tree T

Output: New multicast tree T_{new}

1: **for** the leaving current node v
2: **if** v is a leaf node **then**
3: v sends a leave message to the upstream nodes;
4: **for** node $u \in V(T)$
5: **if** u has receives the leave message **then**
6: $T_{new} \leftarrow T-\{u\}$;
7: **end if**
8: **end for**
9: $T_{new} \leftarrow T-\{v\}$;
10: **end if**
11: **if** v is not a leaf node **then**
12: Mark v as a steiner node;
13: $T_{new} \leftarrow T$;
14: **end if**

15: **end for**
16: **return** T_{new}

Fig. 6. Destination leaving process for GMR-D

When members join or leave a multicast session frequently, the neighbor gradient value between two nodes in the flow network may changes dramatically due to the network states alteration. In detail, there are several reasons for the changes of the neighbor gradient: (1) The neighbor nodes joins/leaves the multicast group. (2) The link load of some links changes. (3) Destination set changes dynamically. The simple but effective strategy for resolving the neighbor gradient dynamically changing problem is to calculate the neighbor gradient periodically. After each specified unit of time, the new multicast tree is pre-built and constructed by the newly discovery route. So the multicast tree can be optimized based on the newest neighbor gradient value. In order to avoid losing data during the process of building the new multicast tree, the data flow will be delivered through the old tree until the new multicast tree is established.

4. Experimental Results

In this section, we test the performance of our proposed algorithms. Our algorithms are implemented in C++ and simulated in multicast routing simulator MCRSIM [25]. We compare GMR-S with two classical multicast algorithms, namely SPT [3] and Jia [7] in the static multicast membership situation. And we also compare GMR-D with WAVE [11] and GMR-S in the dynamic multicast membership situation. WAVE is a source-specific dynamic algorithm which can meet multiple QoS constraints, such as delay and cost simultaneously. Each group of the experiment is repeated 10 times and the average is reported.

Network Topology We use Salama network model [26], which is an improved version of Waxman model [27], to generate network topology. In Salama network generation model, the probability of any pair of nodes (u,v) to be linked is:

$$P_e(u,v) = \frac{k\bar{e}}{|V|} a \exp\frac{-d(u,v)}{bL} \tag{4}$$

where $|V|$ is the total number of nodes in the network, $P_e(u,v)$ is the probability of a directed link from node u to node v, $d(u,v)$ is the Euclidean distance, which is measure by hop count, between node u and node v, b ($0 < b \le 1$) is used to control the number of connections, a ($0 < a \le 1$) control the degree of each node, k is a constant value and $\bar{e}$ is the average node degree of the network. In our experiments, the network topology parameters $a = 0.4$, $b = 0.26$ and $\bar{e} = 4$, $\bar{e} = 8$ are used. **Fig. 7-(a)**, **Fig. 7-(b)**, **Fig. 7-(c)** and **Fig. 7-(d)** illustrate three different size of topological networks with different average node degrees used in this simulation. We note that similar network topologies are also used in many other works [7].

(a) $|V| = 60$, $\bar{e} = 4$ (b) $|V| = 100$, $\bar{e} = 4$ (c) $|V| = 200$, $\bar{e} = 4$

(d) $|V|=60,\ \bar{e}=8$ (e) $|V|=100,\ \bar{e}=8$ (f) $|V|=200,\ \bar{e}=8$

Fig. 7. Network topology used for simulation

As shown in **Table 1**, two types of flows, which are data flow and multimedia flow, are used in this simulation. Similar to [22], we set $\alpha=0.2,\beta=0.4,\gamma=0.4$ for the data flow and $\alpha=0.2,\beta=0.2,\gamma=0.6$ for the multimedia flow.

Table 1. Parameter setting

	Algorithm parameters			**Flow parameters**	
	α	β	γ	Duration(Second)	Payload(kbps)
Data	0.2	0.4	0.4	2-8	20-60
Multimedia	0.2	0.2	0.6	20-80	54-74

Evaluation Metrics: We consider the performance of the compared algorithms in terms of request blocking probability, average hop count and multicast tree cost.

1. *Request Blocking Probability*. The request blocking probability is defined as follows:

$$P_B = \frac{N_{req} - N_{ack}}{N_{req}} \tag{5}$$

where N_{ack} is the number of successful routing requests, and N_{req} is the total number of routing requests.

2. *Average Hop Count*. It is defined as the average hop count from source s to each destination $d \in D$.

3. *Multicast Tree Cost*. The network cost of a multicast tree is defined as the sum of the cost of all links in the tree.

$$C(T) = \sum_{\forall e(u,v)\in T} c(u,v) \tag{6}$$

where $c(u,v)$ is the cost of link $e(u,v)$. Multicast tree cost is a measure of the utilization of the network's resources. For simplicity, we assign the cost of each link is equal to 1 units in our simulation. Hence, the cost of a multicast tree in the network is the number of links in the multicast tree.

4.1 Performance Comparison for Static Multicast Membership

The comparison between GMR-S, SPT and Jia is based on the network topologies presented in **Fig. 7-(a)**, **Fig. 7-(b)**, **Fig. 7-(c)** and **Fig. 7-(d)**. Concretely, the sizes of networks are 60, 100, 200 nodes, and the average node degrees are 4 and 8 respectively. The detailed information about the data flow and multimedia flow are presented in **Table 1**. Moreover, the link capacity is set to 100Mbps for each edge. The group size is varied from 5% to 30% of the total nodes in

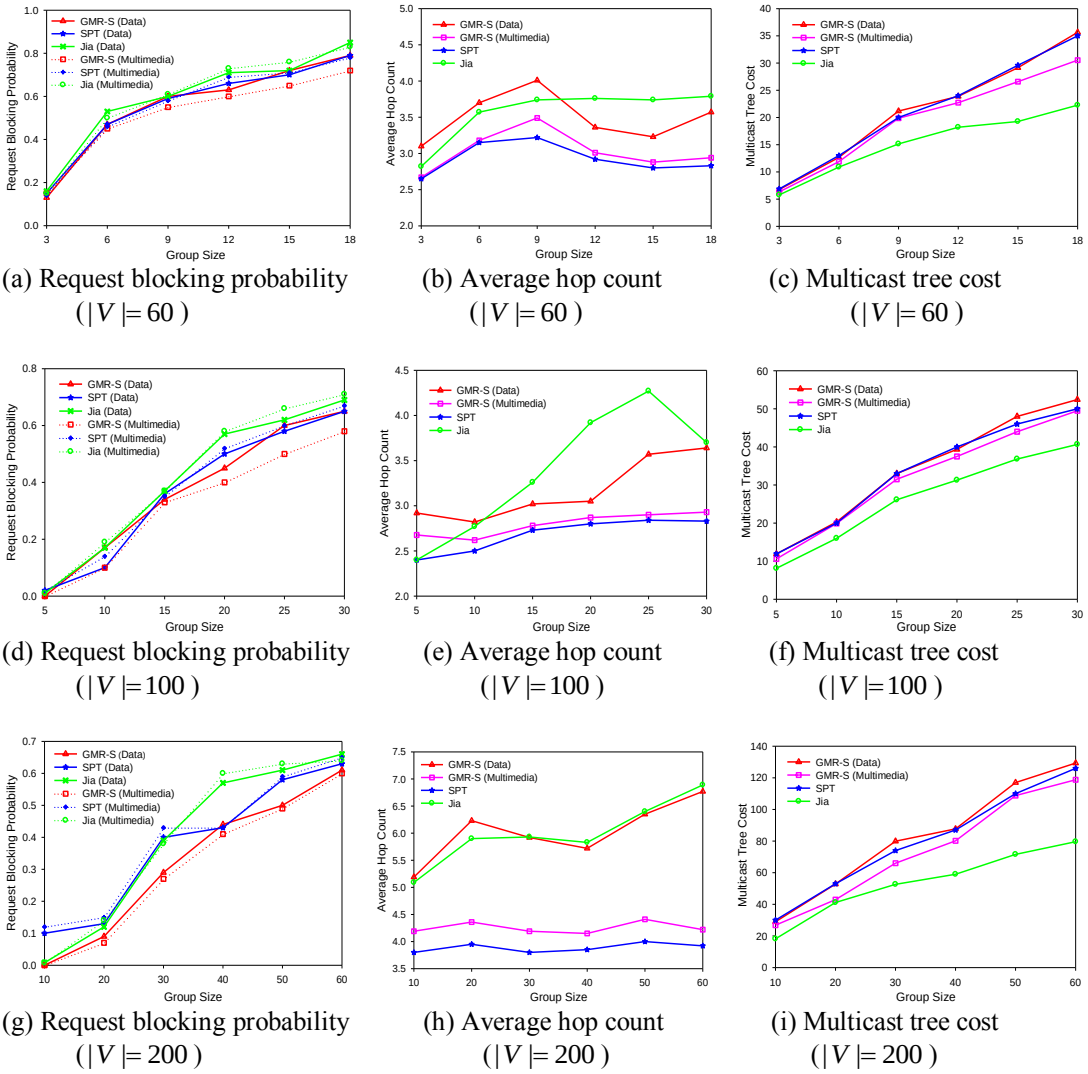
the network. Besides, the request frequency is 100 requests per seconds. The source and destinations are selected randomly.

Fig. 8 exhibits the performance comparison between three algorithms in different size of networks and different type of flows. Since the multicast tree cost and average hop count of SPT and Jia do not affected by the type of flows, not surprisingly, when we discuss the multicast tree cost and average hop count, there is only one curve for both SPT and Jia. However, when we compare the request blocking probability, different compared algorithms will have different performance in different types of flows.

Fig. 8-(a), **Fig. 8-(d)** and **Fig. 8-(g)** illustrate the request blocking probability between three algorithms in different network topologies whose average node degree is 4. We can see that GMR-S has the lowest request blocking probability among all of the compared algorithms in both data flow and multimedia flow. This means GMR-S can establish more multicast sessions than the other two compared algorithms in the similar network situations. That is because GMR-S considers the current load of each link when discovering the route, while SPT and Jia only consider the number of hop count but not the load of each link in the route when discover the routes. From the further observation of **Fig. 8-(a)**, **Fig. 8-(d)** and **Fig. 8-(g)**, we also know that the request blocking probability of the three algorithms gets lower as the size of the network topologies gets larger. Moreover, the advantage of GMR-S is more apparently than SPT and Jia. This further demonstrates GMR-S is able to utilize the spare capacity of the links much more efficiently than the other two algorithms during the route discovery process. Another observation is that, for all of the compared algorithms, the request blocking probability in the data flow is a little higher than the multimedia flow. Furthermore, this gap reduces as the size of the network increases.

Fig. 8-(b), **Fig. 8-(e)** and **Fig. 8-(h)** illustrate the average hop count of different algorithms in different networks. We can see that the average hop count of Jia is the largest in most of the cases, because it does not consider the path length when there is no hop bound. Since SPT calculates the shortest path for each source and destination pair, it achieves the smallest average hop count. We notice that GMR-S also gains the competitive average hop count in both data flow and multimedia flow. That is because GMR-S benefits from the consideration of the hop count for computing the neighbor gradient during each step of the route discovery process. Besides, the appropriate hop count threshold also limits GMR-S generating the long routes.

Fig. 8-(c), **Fig. 8-(f)** and **Fig. 8-(i)** illustrate multicast tree cost of different algorithms in different networks. Since Jia devised the objective, which is only to optimize the network cost, it performs the best on the multicast tree cost among all of the compared algorithms. There is a necessary tradeoff between the request blocking probability and the multicast tree cost for GMR-S. Though GMR-S has the relative higher multicast tree cost than Jia, it still achieves the acceptable multicast tree cost. That is because the higher priority is given to the links, which are already in the multicast tree, during the route discovery process. This leads to more links being shared in the multicast tree. We also notice that the multicast tree cost of GMR-S in multimedia flow is smaller than in data flow. The similar phenomenon is also observed in average hop count. That is because the higher weight of the hop count is provided when calculating the neighbor gradient in multimedia flow than in data flow. Consequently, shorter paths are selected during the route discovery process in multimedia flow than data flow. Despite SPT is easy to get the shortest path for each pair of source and destination, its average multicast tree cost does not have too much advantage than other compared algorithms. That is because SPT only focuses on the shortest path for each pair of source and destination but never considers sharing links of the multicast tree.



(a) Request blocking probability ($|V|=60$) (b) Average hop count ($|V|=60$) (c) Multicast tree cost ($|V|=60$)

(d) Request blocking probability ($|V|=100$) (e) Average hop count ($|V|=100$) (f) Multicast tree cost ($|V|=100$)

(g) Request blocking probability ($|V|=200$) (h) Average hop count ($|V|=200$) (i) Multicast tree cost ($|V|=200$)

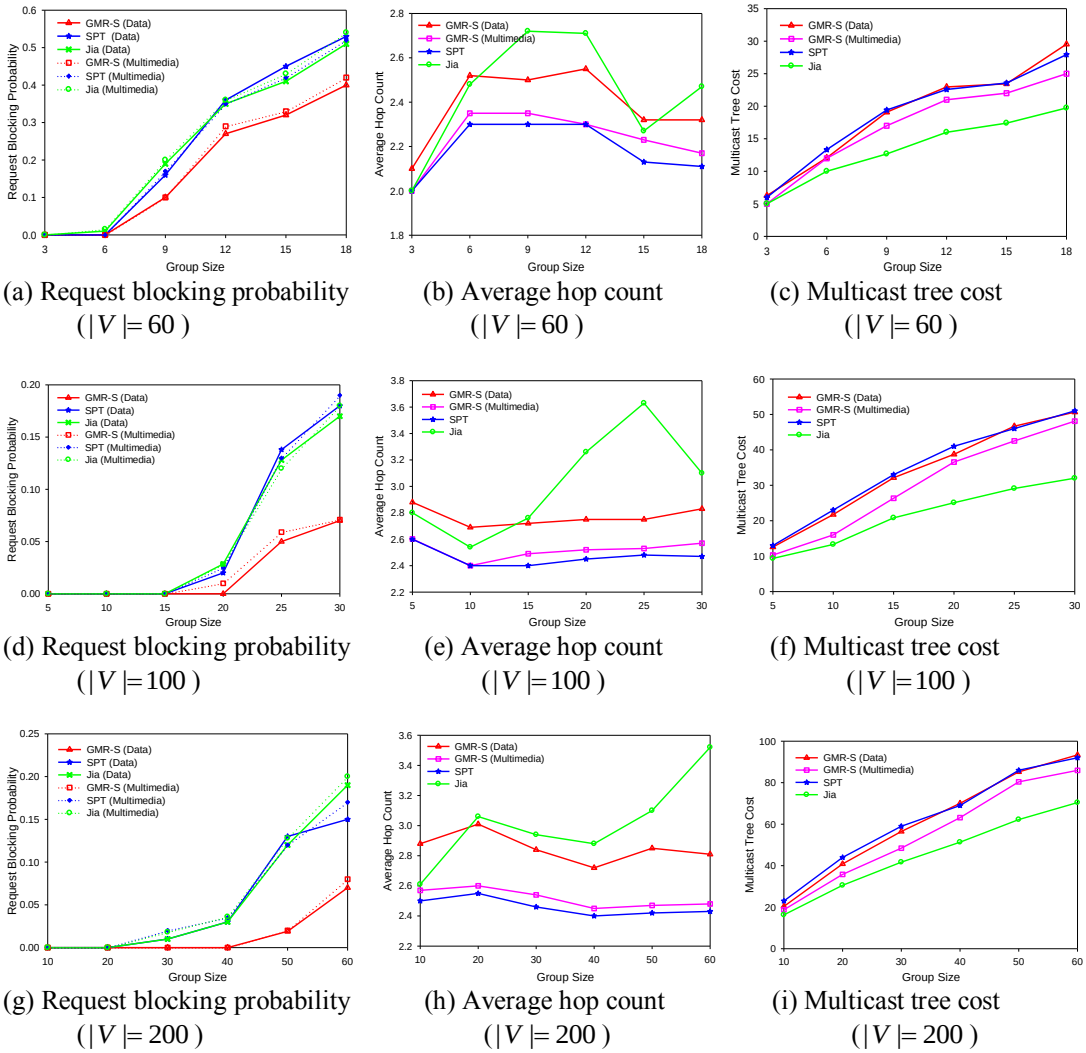
Fig. 8. Performance comparison between GMR-S, SPT and Jia ($\bar{e}=4$)

When the average node degree of the network is 8, the results of the compared algorithms in different size of networks are shown in **Fig. 9**. From **Fig. 9-(a)**, **Fig. 9-(d)** and **Fig. 9-(g)**, we know that, once again, GMR-S outperforms SPT and Jia in the most of cases. Especially, when the group size becomes larger, the advantage of GMR-S is more apparently. Compared with the same size of networks whose average node degree is 4, the request blocking probability of all compared algorithms turns down in networks with the average node degree of 8. That is because the connection probability between any two nodes in the network gets larger as the average node degree increases. Therefore, the probability of successfully constructing multicast trees gets larger when the average node degree is higher.

From **Fig. 9-(b)**, **Fig. 9-(e)** and **Fig. 9-(h)**, we know that the average hop count of GMR-S is better than Jia. That is because GMR-S benefits more from the denser network topologies when discovering the routes. **Fig. 9-(c)**, **Fig. 9-(f)** and **Fig. 9-(i)** give the multicast tree cost of the compared algorithms in networks with average node degree of 8. On the one hand, despite

the multicast tree cost of GMR-S is not as low as Jia, it locates in the very acceptable range. More importantly, the multicast tree cost can be decreased if we give a higher weight to the first attribute of the neighbor gradient during the route discovery process. On the other hand, both SPT and Jia achieve relative low multicast tree cost, but it cannot be adaptive to different types of flows according to different traffic characteristics.

By considering different types of network resources simultaneously, GMR-S can support diverse types of flow more effectively. Moreover, when more multicast trees are constructed in the network, some links may easily be congested. In this case, GMR-S can adjust the discovery route to dynamic network and avoid parts of the network that is highly congested, while SPT and Jia only consider single factor when constructing the multicast tree and cannot be fit for the different requirements of the service-oriented applications.



(a) Request blocking probability ($|V|=60$) (b) Average hop count ($|V|=60$) (c) Multicast tree cost ($|V|=60$)

(d) Request blocking probability ($|V|=100$) (e) Average hop count ($|V|=100$) (f) Multicast tree cost ($|V|=100$)

(g) Request blocking probability ($|V|=200$) (h) Average hop count ($|V|=200$) (i) Multicast tree cost ($|V|=200$)

Fig. 9. Performance comparison between GMR-S, SPT and Jia ($\bar{e}=8$)

4.2 Efficiency for Dynamic Multicast Membership

For a dynamic multicast routing, one of the most important considerations is the tolerance of disturbances caused by frequently group member addition or deletion. For example, in many multimedia services and real-time applications, the frequently changed multicast tree will lead to losing the packets, which arise in flight of the multicast tree. In this case, most of multicast routing algorithms keep the multicast tree unchangeable during the lifetime of the members in the group. Therefore, we use Average Tree Change (ATC), which is also used in [6], to measure the performance of multicast routing for the tolerance of the dynamic changes in the group. ATC is defined as follows:

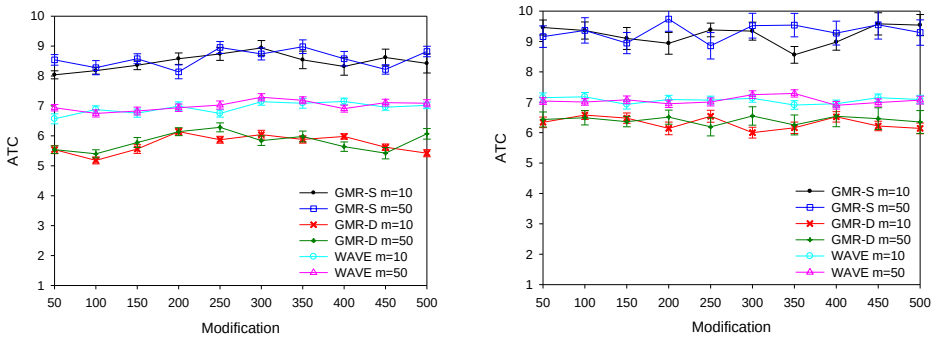
$$ATC = \frac{\sum_{i=1}^{q-1} | (E(T_i) - E(T_{i+1})) \cup (E(T_{i+1}) - E(T_i)) |}{q-1} \tag{7}$$

where q denotes the sequence of requests during the unit time, $E(T_i)$ denotes the edges of the multicast tree at time t. The lower ATC of the compared algorithm means the higher ability for this algorithm to accommodate changes in the group without excessively modifying the multicast tree.

At the same time, we use the function $P_j(m)$, which is introduced by Waxman [27], to generate joining or leaving requests as follows:

$$P_j(m) = \frac{\eta(n-m)}{\eta(n-m)+(1-\eta)m} \tag{8}$$

where n represents the total number of nodes in the network, m is the current number of receivers in the multicast tree, and η is a constant between (0,1). In order to determine the next modification is member joining or leaving process, we generate a random number r ranging from 0 to 1 each time. If $r > P_j(m)$, the modification is leaving and randomly one of the destinations will leave the multicast group. Otherwise, the modification is joining and a node is randomly selected as a new destination.



(a) Multimedia flow (b) Data flow

Fig. 10. ATC Comparison among GMR-S, GMR-D and WAVE ($|V| = 200$, $\bar{e} = 4$)

In order to test the efficiency of GMR-D, we compare GMR-D with GMR-S and a classical dynamic multicast routing algorithm WAVE by metric ATC. In detail, we constantly generate 50 modifications in each unit time. Then, we use the three algorithms to compute the corresponding multicast tree after each modification respectively. Consequently, we compute

the ATC by Equation (7). **Fig.10-(a)** and **Fig.10-(b)** exhibit the changes of ATC among GMR-D, GMR-S and WAVE under different number of modifications. We notice that any change of the group members leads to GMR-S re-computing the multicast tree. However, GMR-D and WAVE can dynamically update the multicast tree for any dynamic change of the members in the group. Not surprisingly, GMR-S has the highest ATC among the three algorithms no matter the group size is 10 or 50 in both multimedia flow and data flow. We also notice that the ATC value of GMR-D does not be affected by the number of modifications and always stays at a low level in both multimedia flow and data flow. This further demonstrates that GMR-D can effectively accommodate the continuous and dynamical changes of the group. Another observation is that, given the same number of modification, the average ATC of both GMR-D and GMR-S in multimedia flow (**Fig.10-(a)**) is a little lower than in data flow (**Fig.10-(b)**). The reason for this phenomenon is that the multicast tree in the data flow has the relative longer path than in multimedia flow. In other words, when a member joins or leaves a group, a relative longer branch will be added to or deleted from the multicast tree. We can also observe that GMR-D performs better than WAVE in both multimedia flow and data flow. Especially, in multimedia flow, the average ATC value of GMR-D is about 20% lower than WAVE. That is because, on the one hand, compared to WAVE, GMR-D always tends to consider the factors of sharing path, residual link capacity and hop count simultaneously when reconstructing the multicast tree. On the other hand, WAVE just considers the cost and delay constraints. When there are changes to the multicast tree, GMR-D can optimize the multicast routing tree more effectively than WAVE.

5. Conclusion

As service-oriented multicast applications become more and more popular, the problem of devising a more adaptive and efficient multicast routing to fulfill the requirements of diverse traffic is an important issue. In this paper, we define a novel neighbor gradient, in which several critical multicast routing factors such as the scale of the multicast tree, link load of the network and hop count of the routing path, are considered simultaneously. Then we presented two distributed multicast routing algorithms, which are GMR-S and GMR-D. GMR-S can be used to generate the multicast tree in the static membership situation, while GMR-D is fit for dynamic membership situation. Experimental results show that our algorithm outperforms SPT and Jia algorithm in terms of request blocking probability with a relative low multicast tree cost and average hop count. Compared to SPT and Jia, our proposed algorithms are more adaptive to different types of flows. Further experiments demonstrate GMR-D can effectively improve the robustness of existing multicast solutions in the situation of dynamic multicast membership.

References

[1] R. Dutta, I. Baldine, A. Wang, M. Iyer and G. Rouskas, "Architectural Support for Internet Evolution and Innovation," in *Proc. of 4th International Symposium on Advanced Networks and Telecommunication Systems*, pp.1-3, Dec.2010. Article (CrossRef Link)

[2] Multimedia research group, Inc. IPTV Global Forecast –2010 to 2014. Available at Article (CrossRef Link)

[3] R. Novak, J. Rugelj and G. Kandus, "Steiner Tree Based Distributed Multicast Routing in Networks," *Steiner Trees in Industries*, vol.11, pp.327-352, 2001. Article (CrossRef Link)

[4] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu and L. Wei, "The PIM architecture for wide-area multicast routing," *IEEE/ACM Transactions on Networking*, vol.4, no.2, pp.153-162, Apr.1996. Article (CrossRef Link)

[5] L. Kou, G. Markowsky and L. Berman, "A fast algorithm for Steiner trees," *Acta Informatica*, vol.15, no.2, pp.141-145, 1981. Article (CrossRef Link)
[6] R. Sriram, G. Manimaran, and C. S. R. Murthy, "A rearrangeable algorithm for the construction of delay-constrained dynamic multicast trees," *IEEE/ACM Transactions on Networking*, vol.7, no.4, pp.514-529, Aug. 1999. Article (CrossRef Link)
[7] X. Jia, "A distributed algorithm of delay-bounded multicast routing for multimedia applications in wide area networks," *IEEE/ACM Transactions on Networking*, vol.6, no.6, pp.828-837, Dec. 1998. Article (CrossRef Link)
[8] T. W. Cho, M. Rabinovich, K. Ramakrishnan, D. Srivastava and Y. Zhang, "Enabling content dissemination using efficient and scalable multicast," in *Proc. of IEEE INFOCOM 2009*, pp.1980-1988, Apr.2009. Article (CrossRef Link)
[9] H. Holbrook and B. Cain, "RFC 4607: Source-Specific Multicast for IP," *Internet Engineering Task Force*, Aug.2006.
[10] L. Guo and I. MATTA, "QDMR: an efficient QoS dependent multicast routing algorithm," in *Proc. of 5th IEEE Real-Time Technology and Applications Symposium, 1999*, pp.213-222, Jun.1999. Article (CrossRef Link)
[11] E. Biersack, and J. Nonnenmacher, "WAVE: A new multicast routing algorithm for static and dynamic multicast groups," in *Proc. of 5th Workshop on Network and Operating System Support for Digital Audio and Video*, pp.228-239, Apr.1995. Article (CrossRef Link)
[12] D. Waitzman, C. Partridge, and S. Deering, "RFC 1175: Distance Vector Multicast Routing Protocol," *Internet Engineering Task Force*, Nov.1988.
[13] G.N. Rouskas, "Multicast Routing with End-to-End Delay and Delay Variation Constraints," *IEEE Journal on Selected Areas in Communications*, vol.15, no.3, pp.346-356, Apr.1997. Article (CrossRef Link)
[14] Y. Yang, J. Wang and M.Yang, "A Service-Centric Multicast Architecture and Routing Protocol," *IEEE Transactions on Parallel and Distributed Systems*, vol.19, no.1, pp.35-51, Jan.2008. Article (CrossRef Link)
[15] M. Yang and Y. Yang, "Constructing minimum cost dynamic multicast trees under delay constraint," in *Proc. of 14th International Conference on Computer Communications and Networks*, pp. 33-138, Oct.2005. Article (CrossRef Link)
[16] D.Yang and W. Liao, "Protocol design for scalable and adaptive multicast for group communications," in *Proc. of IEEE International Conference on Network Protocols*, pp.33-42, Oct. 19-22, 2008. Article (CrossRef Link)
[17] C.C. Wen, C.S. Wu and M.T. Yang, "Hybrid tree based explicit routed multicast for qos supported IPTV Service," in *Proc. of IEEE Global Telecommunications Conference*, pp.1-6, Nov.2009. Article (CrossRef Link)
[18] R. Baumann, S. Heimlicher, V. Lenders and M. May, "HEAT: Scalable Routing in Wireless Mesh Networks Using Temperature Fields," in *Proc. of IEEE International Symposium on a World of Wireless*, pp.1-9, Jun.2007. Article (CrossRef Link)
[19] X. Shi, Y. Chen, G. Lu, B. Deng, X. Li and Z. CHEN, "PMTA: Potential-based Multicast Tree Algorithm with Connectivity Restricted Hosts," in *Proc. of IEEE Global Telecommunications Conference*, pp.559-564, Nov.2007. Article (CrossRef Link)
[20] A. Basu, A. Lin, and S. Ramanathan, "Routing using potentials: A dynamic traffic-aware routing algorithm," in *Proc. of SIGCOMM'03*, pp.37-48, Aug.2003. Article (CrossRef Link)
[21] S. Balasubramaniam, D. Botvich, J. Mineraud, and W. Donnelly, "Parameterised gradient based routing (PGBR) for future internet," in *Proc. of International Conference on Advanced Information Networking and Applications*, pp.58-65, May.2009. Article (CrossRef Link)
[22] S. Balasubramaniam, J. Mineraud, P. McDonagh, P. Perry and L. Murphy, "an evaluation of parameterized gradient based routing with QoE monitoring for multiple IPTV providers," *IEEE Transactions on Broadcasting*, vol.57, no.2, pp.183-194, Jun.2011. Article (CrossRef Link)
[23] Hui Wang, Junfeng He, Tao Li, Shuo Zhang and Zhigang Sun, "heuristic gradient based multicast routing policy for dynamic network," in *Proc. of 2011 International Conference on Multimedia Technology*, pp.4936-4939, Jul.2011. Article (CrossRef Link)
[24] R.W. Floyd, "Algorithm 97: Shortest Path," *Communications of the ACM*, vol.5, no.6, pp.345, Jun.1962. Article (CrossRef Link)
[25] H.F. Salama et al, "MCRSIM simulator source code and Users' Manual," Center for Advanced Computing and Communication, North Carolina State University, Raleigh, 1995.
[26] H.F. Salama, "Multicast routing for real-time communication on high-speed networks [D]," North Carolina State University, 1996.
[27] B.M. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, vol.6, no.9, pp.1617-1622, Dec.1988. Article (CrossRef Link)

Hui Wang received her M.S. degree in Computer Science from National University of Defense Technology, Changsha, China, in 2007. Currently, she is a Ph.D. candidate in National University of Defense Technology. Her research interests include multicast protocol, network architecture and streaming media transmission.

Jianbiao Mao received his B.S. degree in Computer Science from National University of Defense Technology, Changsha, China, in 2010. He is now a M.S. candidate in National University of Defense Technology. His main research interests include computer network protocol and architecture.

Tao Li received his Ph.D. degree in Computer Science from National University of Defense Technology, Changsha, China, in 2010. Now he is an assistant professor. His research interests include network processor, routing and switching.

Zhigang Sun received his Ph.D. degree in Computer Science from National University of Defense Technology, Changsha, China, in 2000. He is now a professor in National University of Defense Technology. His research interests include network communication, routing and switching.

Zhenghu Gong is a professor in National University of Defense Technology. His research interests include network communication and protocol.

Gaofeng Lv received his Ph.D. degree in Computer Science from National University of Defense Technology, Changsha, China, in 2008. Now he is an assistant professor. His research interests include network and information security, routing and switching.