Journal of High Speed Networks 17 (2008) 13–36
DOI 10.3233/JHS-2008-0331
IOS Press

Edge overlay multicast to support comparable multi-class services

Suogang Li [a], Ke Xu [a], Ying Liu [b] and Jianping Wu [a,b]
[a] *Department of Computer Science, Tsinghua University, Beijing 100084, P.R. China*
E-mails: {lsg, xuke}@csnet1.cs.tsinghua.edu.cn
[b] *Network Research Center of Tsinghua University, Beijing 100084, P.R. China*
E-mails: {liuying, jianping}@cernet.edu.cn

Abstract. Traditional IP multicast is likely to imply a huge burden of storage and forwarding for routers. Some issues, such as the scalability problem, will be aggravated when various classes of quality of service are fulfilled in multicast traffic. The recent proposed application layer multicast is more scalable but increases traffic load in network layers and end-to-end delivery delay. In this paper, we present a multicast protocol that supports comparable multi-class services over an overlay network where edge routers are multicast-aware and core routers are oblivious of multicast. The comparable multi-class services consist of different services required by multicast members which can be satisfied via the same delivery tree. Considering the multi-class services and resource limitation on routers, the problem of building minimum cost trees has been proven to be NP-hard. So, we design several feasible heuristic algorithms to solve it. Extensive simulations are conducted to evaluate the performance of the proposed heuristics, and validate the effectiveness of reducing the total tree cost and iteration times under considered constraints. The proposed method is expected to be combined with DiffServ or MPLS VPN networks to fulfill multi-class QoS overlay multicast.

Keywords: Multicast, multi-class services, edge overlay networks, minimum cost trees

1. Introduction

Multicast is an efficient method to support multipoint communications. The native network-layer multicast is efficient and optimal in delivering data to a group of members simultaneously, since it uses a delivery tree to forward data packets only once over each link. However, the network-layer multicast suffers from several problems. The first is difficulty of deployment for it requires all routers in the networks to implement multicast functions. The second is difficulty of supporting quality of service (QoS), whereas multicast multimedia applications need various classes of service, i.e. *heterogeneous QoS requirements*, for each member among the group. The third is the state explosion problem for each group session generates one entry of multicast routing table in each router. This trouble will be aggravated once different QoS classes are provided in the multicast.

Recently, many studies focus on application-layer multicast (ALM) among end systems [1–4] and some of which have actually been realized with peer-to-peer (P2P) technology. The applications of file transfer and video on demand can be achieved on application layer, but the real time applications such as video-conferencing and remote experiment steering need more efficient methods. Although the lack of router support avoids the scalability issue, the application-layer multicast depends on unicasting one or more data copies on network links, therefore it increases traffic in network. Besides, few studies in the literature focus on providing multi-class services in ALM so far due to the difficulty of considering QoS in application layers.

In this paper, our goal is to implement multi-class services for each multicast member in a scalable, efficient and feasible way. Because of the shortage of native IP multicast and ALM, it is reasonable that network layer devices should participate in supporting the demand of multicast and QoS like the way of ALM in the scalable

and deployable fashion so that the Internet Service Provider (ISP) may easily maintain and administer this sort of traffic.

Combining the merits of native IP multicast and ALM, we design an overlay multicast on the edge of the network domain. A domain can be a normal autonomous system (AS). It consists of edge routers (ERs), which act as ingress and egress nodes of the domain and core routers (CRs) lying inside of domain like that of DiffServ [7] network and the provider network specified in BGP/MPLS VPN [10]. We should point out that the state explosion problem in core routers of a domain is more imperative to address than in edge routers. We construct the overlay meshed network on the ERs where multicast is deployed and delivery trees are generated. The CRs may be unaware of multicast therefore achieve good scalability in multicast state. The potential of ERs is exploited so that the edge overlay network is easy to realize and update timely when network is dynamic, since ER knows the topology information of the network layer.

Multi-class QoS requirements that are comparable are supported in the proposed scheme. The comparable services indicate different services that can be required by multicast members through the same tree. Besides, two significant principles are conformed to: one is to reduce total tree cost as much as possible; the other is to be limited by resource of each member router, such as forwarding capability and interface bandwidth. The multicast scheme can be combined with DiffServ or BGP/MPLS VPN networks to fulfill feasible overlay multicast supporting multi-class QoS.

Our multicast mechanism achieves the following targets. First, the total cost of the established delivery tree is as least as possible. Second, not only different QoS classes need to be satisfied among multicast members, but also the limited connectivity of mERs (defined by Definition 2) should be considered, since mERs are the constituent nodes of the overlay meshed network and connect possibly more nodes than it can sustain. We introduce the degree constraint to solve the problem. Lastly, the performance of the established tree is attempted for further improvement. It can be achieved if eligible non-member ERs (nmERs, defined by Definition 2) are invited to join the group. We find the eligible non-member nodes can be used to improve the edge-based overlay multicast through the suitable algorithm described in the subsequent text.

The rest of the paper is organized as follows. The next section reviews related work. Section 3 gives a design overview of the proposed approach. Section 4 describes the problem model. Section 5 proposes three tree-building heuristics in consideration of constraints. Section 6 discusses the proposed algorithm improving tree performance using non-member routers. Section 7 studies the performance of the scheme through simulations. The last section draws a conclusion.

2. Related work

There have been a great number of research papers on QoS multicast in the past decades. At the beginning, many studies focused on solving a theoretical QoS-constrained multicast routing problem, since some of which are NP-hard. Many objectives can be devised and optimized using techniques from combinatorial optimization. Interested readers can refer to three surveys [13–15] and references therein for details. Through the best routing path can be found that by using these methods, computing cost is high and the scalability is not considered.

On the other hand, some researchers take more pragmatic efforts to bring QoS into the existing IP multicast architecture, such as QoSMIC [16], QMRP [17], RIMQoS [18], QoS extension to CBT [19] and PIM-SM QoS extension [20]. QoSMIC uses flooding delivery to find several paths and then the best QoS path is selected. But it results in high communication overhead. So, QMRP finds only one path at first. If QoS requirements are not satisfied in this path, other paths are searched. RIMQoS is based on underlying QoS unicast routing protocol, so the computing cost is low. However, these protocols still use per-flow state and suffer from the state scalability problem.

As discussed above, we focus on implementing multi-class services in multicast. There are some studies concerned with the heterogeneous QoS multicast problem. Some schemes propose to use dedicated multicast groups to carry multimedia information (e.g., video) with different QoS levels to heterogeneous receivers. Receiver-Driven

Layered Multicast (RLM) [21] is a typical example for layered video transmission. Destination Set Grouping [22] uses replicated transmission technology which is similar to layered transmission. But RLM has its own limitations in applications since not all types of multimedia streams can be encoded into layers described above. These approaches avoid maintaining quantitative QoS states in the network, but are difficult to perform traffic aggregation, since QoS classes are exclusively defined and configured by individual external sources.

Multicast with QoS (MQ) [23] is proposed as an integrated framework with consideration of QoS routing, resource reservation and user heterogeneity. This receiver-initiated approach inherits some basic characteristics of RSVP [24]. However, MQ also requires that on-tree routers maintain state on a per-flow basis for end-to-end QoS guarantees, and this aspect still has scalability problem.

Recent studies make an effort to develop multicast in Differentiated Services (DiffServ) [7] to implement scalable QoS multicast. Traffic aggregates based DiffServ is proposed as a promising framework for providing scalable QoS support, while the per-flow based Integrated Services (IntServ) [25] is regarded as being too sophisticated to be implemented and scalable in core routers. However, there are some conflicts between multicast and DiffServ, including the requirements of stateful core vs. stateless core, traffic or QoS required by receivers vs. by sources, the Neglected Reserved Sub-tree (NRS) problem [26], and so on. To tackle these problems, many efforts such as papers of QMD [27], QUASIMODO [28], DAM [29] and DSMCast [30] have been proposed. The schemes of former two need part of core router to support QoS multicast, and the multicast traffic are tunnelled o deliver between these routers. The approach sacrifices the bandwidth efficiency of native multicast. QUASIMODO uses the probe-based admission control method. DAM targets to solve the NRS problem and to accommodate heterogeneous QoS requirements. DSMCast proposes to encapsulate multicast tree information in packet header like Xcast [31]. The scalability is improved in terms of number of groups, however, suffering from the same bandwidth and processing overhead. Through these schemes in above studies, multicast state scalability could be alleviated in some degree, but requires modifying the existing multicast protocol.

EBM [5] and M-DS [6] limit the multicast branching nodes in the ingress edge routers of the domain to scalability problem. This may result in that ingress routers become very busy and interface bandwidth may be not enough. Our proposal considers the resource limitation of the branching nodes.

Several multicast schemes supporting DiffServ-like service in overlay network are proposed recently. DQM [32] is based on the Source Specific Multicast (SSM) [33] model and provides limited qualitative QoS channels for supporting heterogeneous end users. To aggregate traffic with the same QoS in core networks, the authors use the group address in the SSM service model to encode QoS channels, and data packets belonging to the same QoS channel are identified by a common address. MQCT [34] defines the tree QoS cost in the multicast supporting different QoS classes, and designs heuristic to solve the problem of the minimum QoS cost tree. These multicast schemes provide DiffServ-like multi-class services in overlay network without considering the node capability limitation which is an important factor in overlay multicast.

Aggregated multicast [8] is proposed to allow several groups with similar member nodes on the domain to share the same tree, so the state maintained on core routers is decreased greatly. With QoS consideration, the authors then propose AQoSM [35] to fulfill scalable QoS multicast in DiffServ network. The notion of delivering approximate multiple groups in the same tree can mitigate the state scalability problem in the backbone routers. QMCT [9] employs the notion and designs the scalable QoS multicast scheme. Obviously, the aggregating methods trade some wasted bandwidth off the reduction of state maintenance in routers.

In overlay multicast or application layer multicast (ALM) [1–4], no router is needed to support multicast; end hosts accomplish member management and multicast forwarding instead. Being unable to support multi-class QoS requirements, TAG [3] allows the hosts to learn network topology to optimize application multicast trees. MSN [4] considers interface bandwidth limitation in tree building in a way imposing degree constraints on each multicast node. Recent studies of ALM in [36] and [37] only consider the member priority or only concern with simple QoS support. Besides, OverQoS [38] offers a manner to enhance the best-effort service on the basis of overlay networks. It uses the controlled loss virtual link (CLVL) abstraction to limit the loss rate observed by an aggregate of traffic. OverQoS can simultaneously provide both statistical loss guarantees and bandwidth guarantees at the cost of a low bandwidth overhead and bounded end-to-end delay. The overlay multicast mechanism may utilize the OverQoS to support QoS.

To the best of our knowledge, there is no literature considering both the QoS classes and the node capacity when building delivery tree in overlay network so far. This paper will discuss the problem.

3. Design overview

3.1. Definitions

The current Internet includes many network domains. We first give the definitions of them.

Definition 1 (*Network domain*). Generally, a network domain comprises the edge and the interior, including ERs and CRs, respectively.

A network domain can be a normal autonomous system (AS). With little state information on packet flows, the CRs are only in charge of forwarding packets as quickly as possible, whereas the ERs offer necessary functions for flows. The functions include resource reservation, flow control, packet marking, VPNs management, tunnels, multicast, etc.

Definition 2 (*Member edge routers*). The ERs of a network domain are multicast member ER, called mER(g), when a group session g transits it. Correspondingly, the other ERs which the group session does not transit are called non-member ERs, i.e., nmER(g).

The mER is called source ER (sER) or ingress ER (ingER) where the session enters the domain. The mER are called the receiver ER (rER) or egress ER (egER) where the multicast group leaves the domain.

Different QoS requirements are supported in the proposal. It is important to present the comparability between QoS classes. The definition is shown as follows.

Definition 3 (*Comparable QoS classes*). Given two classes of QoS requirements, noted as q_1 and q_2, if the service satisfying q_1 can also satisfy q_2 while satisfying q_2 cannot satisfy q_1, then we say q_1 and q_2 are comparable, denoted by $q_1 \succ q_2$ or $q_2 \prec q_1$.

For example, PS (Premium Service) and AS (Assured Service) are defined in DiffServ model. In contrast with the best-effort (BE) traffic, the based service in network layer, PS $\succ$ BE, AS $\succ$ BE. However, it is not comparable between the service classes of PS and AS. The reason some QoS classes are incomparable is that the metrics of these QoS classes are different, e.g., we cannot determine which is higher between a little delay service and a high loss service. The case of incomparable classes will be studied in future.

We focus on comparable QoS classes in overlay multicast. It means that the node requiring high class QoS should not be placed downstream of the node requiring low class QoS in a delivery tree if these two nodes are members of the same multicast session. We will study how to arrange the members according to the QoS class required by each member. Note that how to implement certain QoS class in a multicast node exceeds the scope of discussion in this paper, since it can be achieved through the node scheduling mechanism and buffer management, just as achieving QoS in unicast.

3.2. Edge overlay QoS multicast

The proposed QoS multicast scheme pushes the multicast function out to the network edges. All ERs in the domain organize an overlay meshed network. On the edge overlay network, a tree with the minimum cost is constructed to connect all the mERs related with the multicast group. The scheme is called edge overlay QoS multicast (EOQM), which takes advantage of edge routers of the domain and combines the merits of both application-layer

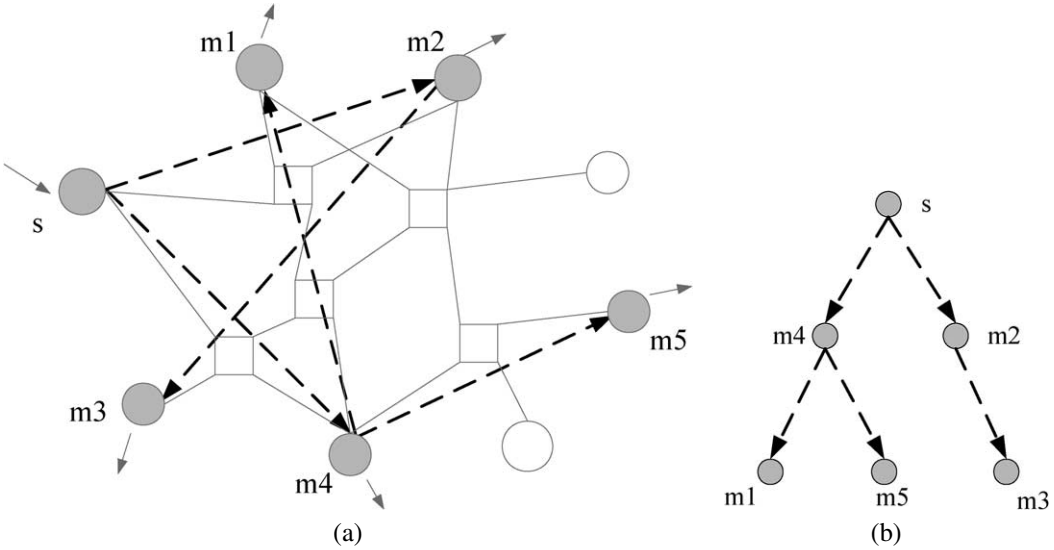

Fig. 1. An instance for EOQM.

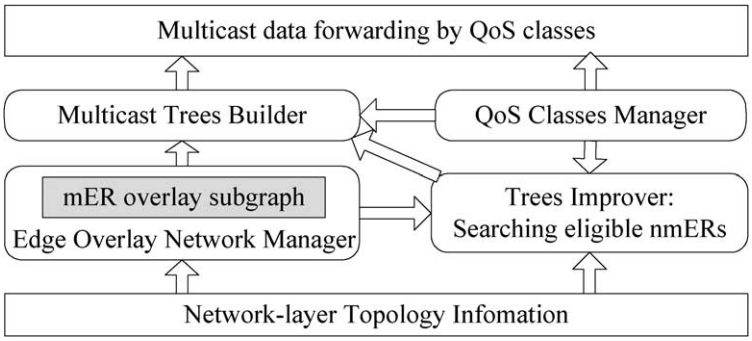

Fig. 2. EOQM function components on ingERs.

and native network-layer multicast. Therefore, it becomes convenient for ISPs to deploy and manage QoS multicast services and utilize network resource efficiently. The CRs are completely free from maintaining any multicast state.

Figure 1 shows an instance of the edge overlay QoS multicast. The square nodes denote the CRs, and the round nodes denote the ERs. The gray round nodes are mERs of a group with ingress routers. The solid lines denote the links between nodes, and the dashed lines denote the membership among the member ERs. Figure 1(b) shows the multicast tree for the group on the overlay network of complete ERs.

We assume that the ingress ER of each multicast session is designed for computing the multicast tree. The ingERs have full knowledge of other ERs in the session, and compute the tree according to the current conditions of the available bandwidth. The function components of the ingER are shown in Fig. 2. In this figure, the ingERs utilize directly the network layer topology information to compute multicast routing trees. The components of tree building and improvement will be discussed in the next two sections, respectively.

3.3. Requirements and targets

An advantage of EOQM is that it does not need routing state updates via broadcasting among all nodes, whereas the overlay multicast needs to update the state periodically since the participant nodes are not routers but hosts.

Therefore, routing performance of the later may be deterred due to the imprecise routing information [4].

In the routing of EOQM, ingERs have full knowledge of other ERs in the session through existing routing protocols, e.g., OSPF. It is able to compute the tree according to its current conditions of the available bandwidth. We should point out that the centralized route computation is used to increase routing efficiency and reduce message complexity. The centralized version does not create a single point of failure or even a performance bottleneck, as each session may select its ingress ER to perform the tree computation. This means that the overall computational load can be inherently distributed among different ERs for different sessions.

Four targets are achieved in the multicast scheme as follows:

(i) The total cost of multicast tree spanning all mERs is as least as possible.
(ii) Heterogeneous QoS classes required by mERs are to be supported.
(iii) The mERs should be subjected to the constraint of resource limitation.
(iv) Eligible nmERs are introduced into the group to improve the tree performance.

Target (i) is aimed to reduce the expenditure of multicast forwarding in the domain. The cost of the path between mERs is acquired by network layer information. For target (ii), the scheme assures that the node requiring high class QoS should not be placed downstream of the node requiring low class QoS in the tree. It is referred to as the *multi-class QoS constraint* (*MQC*). Target (iii) indicates that another restriction is supposed to be satisfied, known as the *member resource constraint* (*MRC*), which is related with the power of forwarding and the access capacity of interface in the mERs, and decides the number of children a tree node can have.

3.4. An application instance in BGP/MPLS VPN

The service of BGP/MPLS VPN is specified in [10]. ISP provides VPNs (Virtual Private Networks) to the customers by using its network domain consisting of P (Provider) routers and PE (Provider Edge) routers. The customers connect the PE through CE (Customer Edge) routers. The MPLS technology is used to implement VPNs in the P network domain. Herein, the PE router is an ER; the P router is a CR. Our proposal can be realized in the network model. The draft [11] specifies the multicast VPNs in the BGP/MPLS networks, which can support several scalable multicast schemes but cannot offer the heterogeneous QoS requirement among the multicast members. Again, the scheme tunneling the multicast packets from the ingress PE to all of egress PEs may make the ingress PE bear heavy load when many egress PEs are involved in the particular group session. These disadvantages can be compensated if applying the EOQM.

4. Problem formulation

An edge overlay network (EON) is a fully connected virtual network formed by edge routers which communicate with each other using links in the domain. The EON is modeled by a complete graph $G = (V, E)$, where V is a set of nodes representing ERs, and E is a set of edges. Each edge, being undirected, represents a unicast path between two ERs. Each edge $e(u, v) \in E$ is associated with a cost, $C(e)$, which is defined as the cost of shortest path from u to v obtained according to current routing state information. It represents the expenditure of passing the edge.

Multicast services are offered in the EON. All the mERs of a session g in a domain compose a node subset, $M(g) \subseteq V$. The ingress node is s. The multicast tree of g is a directed tree rooted at s, called $T(s \rightarrow M(g))$ or $T(s \rightarrow M), T \subseteq G$. The total cost of T is $C(T) = \sum_e C(e)$. In the graph G, we call the overlay graph only formed by nodes in M a *pure* subgraph of g, denoted $G^P(g)$. Combining one or more non member nodes with $G^P(g)$, the overlay graph is called a hybrid subgraph, called $G^H(g)$.

The multicast tree built is needed to satisfy the heterogeneous QoS requirement. The available QoS class set is Q. The size of Q is denoted by Q^N. The multicast member $v \in M$ requests QoS class $q(v) \in Q$. The constraint of MQC indicates the directed transfer from high class QoS node to low class QoS node. That is, the direction of edge e is $u \rightarrow v (e : u \rightarrow v)$ if $q(u) > q(v)$. Obviously, the QoS class of source s is the highest, i.e., $q(s) = 0$.

The limitation of MRC corresponds to the degree restriction of the nodes in the graph. Each node v in the graph is attached with a degree limitation $D^{\text{Max}}(v)$. The degree of node $v \in T$ is denoted as $d_T(v), d_T(v) \leqslant D^{\text{Max}}(v)$. For each node v, $D^{\text{Max}}(v) \geqslant 1$, if node v is on the tree.

The cost of the built multicast tree is expected to be as low as possible, as well as satisfying the constraints of MQC and MRC. We assume that the QoS requirement of each node can be met without considering resource reservation problem in our study. We refer to the problem as computing minimum cost trees with constraints of QoS class and node degree, QD-MCT for short. The definition is given as below.

Definition 4 (*Minimum cost directed tree problem (QD-MCT) with constraints of QoS class and node degree*). Given an undirected complete graph $G(V, E)$, a non-negative real cost $C(e)$ for each edge, a positive degree limitation $D^{\text{Max}}(v) \in Z^+$ and a QoS class $q(v) \in Q$ for each node v, where Q is the set of QoS classes supported, $Q = \{0, 1, \ldots\} \subset \text{Z}$ with sort descending, the size of which is denoted by Q^N; find a spanning tree T of $M \subseteq V$ of minimum cost, subjecting to the constraints that the node u requiring high QoS class should not be placed downstream of the node v requiring low QoS class in T, for all $u, v \in M$ and that the node v degree in T is less than $D^{\text{Max}}(v)$ for all $v \in M$.

This problem can be formalized as

$$\begin{aligned}
&\text{Minimize} \quad C(T(s \to M)) = \sum_{e \in T} C(e) \\
&\text{Subject to} \quad \text{(i) } \forall e(u, v) \in E_T, e : u \to v, \text{ if } q(u) > q(v); \\
&\qquad\qquad \text{(ii) } d_T(v) \leqslant D^{\text{Max}}(v).
\end{aligned}$$

The optimal solution of the QD-MCT problem is denoted as T^{opt}, and the total cost of T^{opt} is denoted as $C(T^{\text{opt}})$.

Theorem 1. *The QD-MCT problem is NP-hard.*

Proof. Solving the minimum cost tree covering the node subset M in a graph is a well-known NP-hard problem of Steiner Minimum Tree (SMT). The QD-MCT problem involves the constraints of MQC and MRC, which are corresponded to the problem of computing directed SMT with degree limitations. Especially, when $D^{\text{Max}}(v) = 2$ for all $v \in M$, the problem is the same as a Hamilton Path problem, which is NP-hard [39]. We transform from the Hamilton Path problem for the general case of $D^{\text{Max}}(v) \geqslant 2$. Let $G = (V; E)$ be the graph of a Hamilton Path problem instance. We transform G to $G_1 = (V_1, E_1)$ by adding $D^{\text{Max}}(v) - 2$ nodes $u_1, \ldots, u_{D^{\text{Max}}(v)-2}$ to each $v \in V$, and join each of these new nodes u_i to v with edges of zero length. All the other edges from u_i have infinite length. Now the QD-MCT instance in G_1 has a spanning tree if and only if the Hamilton Path instance has a path of that which includes all the nodes in G. □

5. Algorithms to build trees

We focus on how to design the heuristic algorithms to solve the QD-MCT problem, i.e., to compute the minimum cost spanning trees with constraints of MQC and MRC in the overlay pure subgraph $G^P(g)$ since it is NP-hard. First, we consider the case of no degree limitation.

5.1. Tree-building algorithm without degree limitation

If only considering multi-class QoS constraint, the problem turns into solving the minimum cost directed tree with MQC in $G^P(g)$, called *Q-MCT* problem. To solve the problem, Q-MCT algorithm is proposed. As we all

Algorithm 1
Q-MCT algorithm

Input: The graph, $G^P(g) \subseteq G$; source node, s; node $v \in V$ requiring QoS class $q(v)$, $q(s) = 0$. The QoS number, Q^N. The smaller q, the higher QoS class is. Edge cost, $C(e(u, v))$ for $e(u, v) \in E$.

Output: Tree $T(s)$

(0) $T = \varnothing$ and push s into T.

(1) Classify the rest nodes into $R_i, i \in [0, Q^N - 1]$, according to the QoS class of each node;

(2) **for** i **from** 0 **to** $Q^N - 1$,

(3) **while** $(R_i \neq \varnothing)$, **do**

(4) Select the node u in R_i to satisfy formula (1);

(5) Add the node u and $e(u, v)$ in T and set $e : u \to v$;

(6) Remove the u from R_i.

know, using the well-known Prim's algorithm can compute Minimum Spanning Trees (MST) of a graph in polynomial time [12]. The Q-MCT algorithm builds a spanning tree of $G^P(g)$ incrementally, like the Prim's algorithm.

In Prim's algorithm, a new edge with its connecting node off-tree is selected to join the current built tree if the edge cost is the least in all candidate edges and results in no loop in the current tree. The steps go on until all nodes are contained. In Q-MDT algorithm, a variant $len_T(u)$ is maintained for each off-tree node $u \in M - T$, which is denoted the closest distance between node u, current tree T, and $len_T(u) = \min_{v \in T}\{C(e(u, v))\}$. The algorithm prefers to select the node with minimum len_T, i.e.,

$$\min_{u \in V-T}\{len_T(u)\} = \min_{u \in V-T}\Big\{\min_{v \in T}\{C(e(u, v))\}\Big\}. \tag{1}$$

In the algorithm, we prefer to select the nodes with high rather than low class QoS to join the current subtree. Therefore, the algorithm satisfies that the class QoS of every node on the current tree is not lower than that of any candidate node off the tree, and the direction of the newly selected edge is from the end on tree to the end off tree before joining. Finally, the gained tree is guaranteed to satisfy the MQC. In the initialization of the algorithm, the sER must be the first to join the tree for its default highest class QoS. It is assumed that the edges which cannot satisfy the node QoS requirement are omitted from $G^P(g)$ before the algorithm starts up. See Algorithm 1.

The difference between the Q-MCT and the Prim's algorithms is node classification in step (1). If there are n nodes and m edges in $G^P(g)$, the Prim's algorithm has a time complexity of $O(m \log n + n \log n)$ using ordinary binary heaps. By using Fibonacci heaps [12], it can be sped up to run in time $O(m + n \log n)$, which is an improvement since an overlay graph is a fully connected graph and $m = O(n^2)$. Node classification in step (1) of the Q-MCT algorithm needs to run in time $O(n)$. The later steps are like the Prim's algorithm. So the total running time of the algorithm is $O(n + m + n \log n)$ if using Fibonacci heaps.

Theorem 2. *The result of the Q-MCT algorithm is a minimum cost multicast tree with MQC in $G^P(g)$.*

Proof. First, the result T of the Q-MCT algorithm is spanning tree for $G^P(g)$. The reason is that at beginning the node s is in the tree T. And in the loop part of steps (2)–(6), one node and one edge are selected to join T each time, which runs $n - 1$. As a result, there are n nodes and $n - 1$ edges, which is a spanning tree. Next, the tree T is of minimum cost. Because each node joins the tree in the descending order of its QoS classes (from 0 to $Q^N - 1$), the all directed input edges (i.e., the edges from other nodes to the node) of the candidate node to join the tree are checked, and the selected node is the closest to the tree each iteration. Therefore, the final tree is of minimum cost. □

In Q-MCT algorithm, if the joining order of QoS class is not conformed to, the selected node is possibly not the closest to the current tree as Fig. 3 shows. The graph is shown in Fig. 3(a). The figures in the parenthesis to the right of a node are the QoS classes by the node. The optimal tree is shown in Fig. 3(b) and non-optimal result is shown in Fig. 3(c) not in the order of QoS classes.

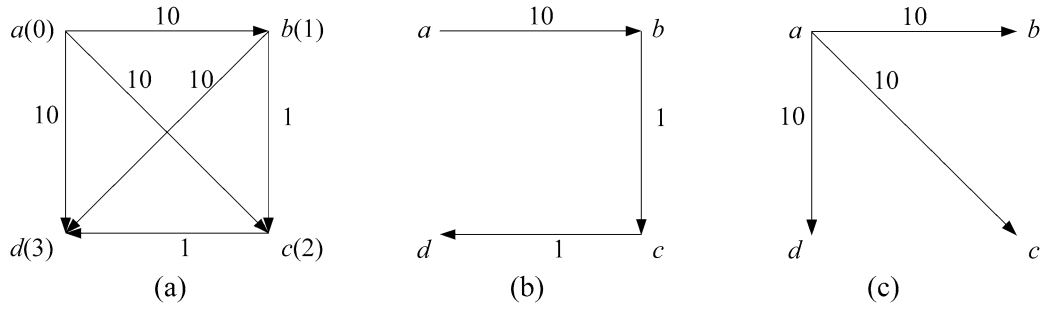

Fig. 3. Q-MCT algorithm in the different orders: (a) a graph G; (b) $C(T^{\text{opt}}) = 12$, in the order of b–c–d; (c) $C(T_1) = 30$, in the order of d–c–b.

Algorithm 2

QD-Heuristic for QD-MCT

Input: The graph, $G^P(g) \subseteq G$; source node, s; node degree limitation, $D^{\text{Max}}(v)$ and required QoS class, $q(v)$, for $v \in V$, $q(s) = 0$. The QoS number, Q^N. Edge cost, $C(e(u, v))$ for $e(u, v) \in E$.

Output: Tree $T(s)$

(0) $T = \varnothing$ and push s into T.

(1) Classify the rest nodes into $R_i, i \in [0, Q^N - 1]$, according to the QoS class of each node;

(2) **for** i **from** 0 **to** $Q^N - 1$,

(3) **while** $(R_i \neq \varnothing)$, **do**

(4) Select the node u in R_i to satisfy formula (2);

(5) Add the node u and $e(u, v)$ in T and set $e : u \rightarrow v$;

(6) Remove the u from R_i and update u and v degrees.

5.2. Heuristic algorithm with degree limitation

For the MRC, assuming the degree limitation of each node is D^{Max}, the algorithm should select the closest off-tree node, u, to the current tree T. And its adjacent node on the tree, v, must comply with $d_T(v) < D^{\text{Max}}(v)$. If not, another candidate edge with the second least cost is considered to join. That is,

$$\min_{u \in V-T} \left\{ \min_{v \in T} \{C(e(u, v))\} | d_T(v) < D^{\text{Max}}(v) \right\}. \tag{2}$$

We call the algorithm for QD-MCT problem QD-Heuristic, as shown in Algorithm 2. The edges which cannot satisfy the node QoS requirement are omitted from $G^P(g)$ before the algorithm starts up.

The running time of QD-Heuristic is the same with the Q-MCT algorithm, totally $O(n + m + n \log n)$ using Fibonacci heaps, though the degree limitation is executed in step (4).

Let $C_{\text{max}} = \max_{e \in M}\{C(e)\}$, $C_{\text{min}} = \min_{e \in M}\{C(e)\}$. The optimal solution of the QD-MCT problem is $C(T^{\text{opt}}) > nC_{\text{min}}$. If we denote the solved tree of QD-Heuristic as T^{H} and the tree cost as $C(T^{\text{H}})$, then $C(T^{\text{H}}) < nC_{\text{max}}$. The performance approximation ratio of QD-Heuristic is $C(T^{\text{H}})/C(T^{\text{opt}}) < C_{\text{max}}/C_{\text{min}}$.

For degree limitation, the QD-Heuristic cannot assure to compute the optimal result. For example, the graph is shown in Fig. 4(a). The figures in the brackets to the right of a node are the QoS classes by the node. The result computed by QD-Heuristic is shown in Fig. 4(b) and the real optimal result is shown in Fig. 4(c). The reason is that after b and c joining the tree, the degree of a is exhausted for its degree limitation is 2.

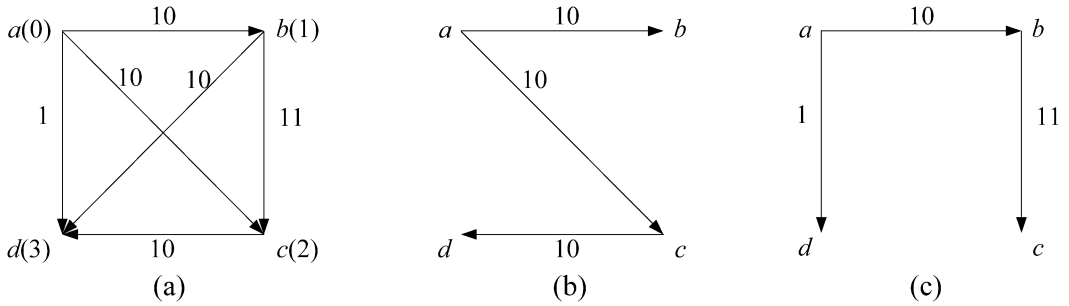

Fig. 4. An example for the QD-Heuristic with each node degree limitation of 2: (a) a graph G; (b) $C(T_{\mathrm{A}}) = 30$ by QD-Heuristic; (c) $C(T^{\mathrm{opt}}) = 22$, the optimal solution.

Algorithm 3
DH algorithm for QD-MCT

Input: The graph, $G^P(g) \subseteq G$; source node, s; node degree limitation, $D^{\mathrm{Max}}(v)$ and required QoS class, $q(v)$, for $v \in M, q(s) = 0$. The QoS number, Q^N. Edge cost, $C(e(u, v))$ for $e(u, v) \in E$. k is initialized.

Output: Tree $T(s)$

(0) $T = \varnothing$ and push s into $T, d_T(s) = 1, d_T(v) = 0, \forall v \in V - \{s\}$;

(1) Classify the rest nodes into $R_i, i \in [0, Q^N - 1]$, according to the QoS class of each node;

(2) **for** i **from** 0 **to** $Q^N - 1$,

(3) **while** $(R_i \neq \varnothing)$, **do**

(4) Select k nodes (if existing) to compose set K; then select node $u \in K$ to satisfy formula (3);

(5) Add the node u and $e(u, v)$ in T and set $e : u \rightarrow v$;

(6) Remove the u from R_i and update u and v degrees.

5.3. Degree-based heuristic

Enlightened from the example of Fig. 4, when we select nodes to join the tree in the QD-Heuristic, we make the residual degree of the related nodes as large as possible and let subsequent nodes have an opportunity of becoming the closest candidate to the tree. In MSN [4], the similar idea is applied.

We define the *residual degree*, $d_R(i)$ of node $i \in V$ as $D^{\mathrm{Max}}(i) - d_T(i)$ and $d_R(i) = D^{\mathrm{Max}}(i)$ if i is not on the tree T. We modify the step (4) in the QD-Heuristic: to select k off-tree nodes (if existing) that are closest to the tree to compose the *candidate set* $K \subseteq M - T$, and then to select a node $u \in K$ that maximizes the smaller one from $d_R(u)$ and $d_R(v)$, i.e.,

$$\max_{u \in K, v \in T} \Big\{ \min(d_R(u), d_R(v)) | d_T(v) < D^{\mathrm{Max}}(v) \Big\}. \tag{3}$$

We call the algorithm Degree–Heuristic (DH) as QD-MCT problem, shown in Algorithm 3. The edges which cannot satisfy the node QoS requirement are omitted from $G^P(g)$ before the algorithm starts up.

The difference between the DH algorithm and the QD-Heuristic is in step (4), where k off-tree nodes (if existing) are selected to compose the candidate set K, which requires the time of $\mathrm{O}(k \log n)$. Then the node with the largest residual degree is selected to join the tree, which requires the time of $\mathrm{O}(k \log k)$. The total time of step (4) is n. Therefore, the total running time of the DH algorithm is $\mathrm{O}(n + m + kn(\log n + \log k)) = \mathrm{O}(n + m + kn \log n)$.

5.4. Cluster-based heuristic

In each iteration of both QD-Heuristic and DH algorithms, only node is selected to join the tree. In order to reduce the iteration times for building the tree, we design an algorithm based on clusters, called Cluster–Heuristic (CH). The ECT [5] uses similar idea in a simple fashion, but the node degree limitation is not considered.

So the forwarding burden in ingress router is huge and the shape of the tree is short and fat. The proposed CH algorithm assures each node in the tree comply with its degree limitation. The difference between the CH and QD-Heuristic algorithms is that CH selects not only one node but one cluster of nodes to join the tree each time. Three unique steps are included in CH algorithm, as follows.

(1) *Cluster construction*. In the overlay network, a cluster *Clu*(u) is centralized on node $u \in M - \{s\}$, and involves *Nc*(u) neighbor nodes of u. Node u is called cluster core and the *Nc*(u) neighbors are called cluster leaves. The size of cluster *Clu*(u) is *Nc*(u), $0 \leqslant Nc(u) \leqslant \lceil \alpha D^{\text{Max}}(u) \rceil$, where $0 < \alpha < 1$, and *Nc*(u) must be smaller than $D^{\text{Max}}(u)$. The conditions where node x becomes the leaf of cluster centralized in u are: it is the closest *Nc* neighbor to u; its QoS class is not higher than u, i.e., $q(x) \leqslant q(u)$. The cost of cluster *Clu*(u) is defined as an average of all costs from cluster core to cluster leaves, namely,

$$C(\textit{Clu}(u)) = \frac{\sum_{x \in \textit{Clu}(u)} C(e(u \to x))}{\textit{Nc}(u)}. \tag{4}$$

(2) *Cluster join*. Each time, a cluster is selected to join the tree in the order of descending QoS classes (from 0 to $Q^N - 1$). In the algorithm, a variant $\textit{lsum}_T(\textit{Clu}(u))$ is maintained for each cluster *Clu*(u), which is defined as sum of the closest distance from the cluster core to the current tree and the cluster cost, i.e., $\textit{lsum}_T(\textit{Clu}(u)) = \min_{v \in T}\{C(e(u,v))\} + C(\textit{Clu}(u))$. The algorithm selects a cluster *Clu*(u) that minimizes $\textit{lsum}_T(\textit{Clu}(u))$ and satisfies the degree limitation, which is shown as below:

$$\min_{u \in V-T}\{\textit{lsum}_T(\textit{Clu}(u))\} = \min_{u \in V-T}\Big\{\min_{v \in T}\{C(e(u,v))\} + C(\textit{Clu}(u)) | d_T(v) < D^{\text{Max}}(v)\Big\}. \tag{5}$$

The cluster *Clu*(u) joining the tree indicates that all the core and leaves included in *Clu*(u) are on the tree and the core is the common parent node of the leaves.

(3) *Cluster update*. After the *Clu*(x) joining the tree, any off-tree cluster *Clu*(y) including the core or leaf node of the *Clu*(x) will delete these nodes and replace them with other off-tree nodes if existing. The replacement prefers closest distance nodes first. And then, the cost and size of *Clu*(y) are updated.

The detail process is shown as Algorithm 4. Before it starts up, the edges which cannot satisfy the node QoS requirement are omitted from $G^P(g)$.

The CH algorithm builds the tree more quickly since it selects a cluster of nodes instead of one node to join the tree in each iteration. In the algorithm, the cluster size *Nc*(u) is set lower than the degree limitation of cluster core $D^{\text{Max}}(u)$ so that the node u after joining the tree also has an opportunity to act as the parent of other off-tree nodes.

Algorithm 4
CH algorithm for QD-MCT

Input: The graph, $G^P(g) \subseteq G$; source node, s; node degree limitation, $D^{\text{Max}}(v)$ and required QoS class, $q(v)$, for $v \in M, q(s) = 0$. The QoS number, Q^N. Edge cost, $C(e(u,v))$ for $e(u,v) \in E$. α is initialized for cluster sizes.

Output: Tree $T(s)$

(0) $T = \{s\}, d_T(s) = 1, d_T(v) = 0, \forall v \in M - \{s\}, n_T = 0$;

(1) For each node $v \in V - \{s\}$, construct *Clu*(v) and account the cost using formula (4);

(2) Classify the clusters into $C_q, q \in [0, Q^N - 1]$, according to the QoS class of core;

(3) **while** $n_T \neq n$, **do**

(4) **for** each C_q (q from 0 **to** $Q^N - 1$), **do**

(5) **while** ($C_q \neq \varnothing$), **do**

(6) Select cluster *Clu*(u) to satisfy formula (5);

(7) Add *Clu*(u) in T, update $n_T \mathrel{+}= (Nc(v) + 1)$, increase the degrees of u and cluster nodes;

(8) Remove the *Clu*(u) from C_q;

(9) Examine each off-tree clusters, delete the nodes included in the *Clu*(u), replace with other off-tree nodes and update the cluster cost and size.

If we denote the number of clusters as C_n, then $C_n = \theta n$, $0 < \theta \leqslant 1$. Specially, $\theta = 1/Nc$ and $C_n = n/Nc$ when each cluster size is identical. In step (1) of the CH algorithm, the running time of cluster construction is $Nc \cdot n \log n$. In step (2), node classification spends the time of C_n. In each iteration, step (6) needs the time of $O(\log n)$ and step (9) needs the time of $O(C_n \cdot Nc) = O(n)$. For the iteration is executed C_n times, the entire iteration part needs the time of $O(C_n \log n + nC_n)$. Therefore, the total running time of the CH algorithm is $O(m + Nc \cdot n \log n + C_n \log C_n + nC_n) = O(m + Nc \cdot n \log n + \theta n \log \theta n + \theta n^2)$.

6. Improving trees through non-member routers

After presenting the principles of the proposed multicast scheme and the tree-building algorithm, we study how to improve the minimum cost tree in the section. The idea is to search for some eligible nmER nodes, called *improving nodes*, which join the multicast tree in order to reduce the total tree cost. We mainly discuss the improvement of the tree deduced from QD-MCT of Algorithm 2 and the trees from other algorithms can be analogously improved using the idea. Above all, we explain the possibility of improving the existing nodes.

6.1. Chances of improvement

A general case of tree improving is to be explained here. Figure 5(a) shows a part of network topology. The relevant overlay network shown in Fig. 5(b) comprises nodes o and $m_1 \sim m_n$. Node o connects $m_1 \sim m_n$ through x_u, and then $y_1 \sim y_n$, respectively. And the cost of path ox_u, x_uy_i and y_im_i is p_o, u_i and $l_i, 1 \leqslant i \leqslant n$, respectively. Node r joins the overlay network as shown in Fig. 5(c), where r connects other nodes by x_v, the cost of path rx_v, x_ux_v and x_uy_i is p_r, p_x and v_i, respectively. We assume the cost of all paths is positive.

In the overlay network of Fig. 5(b), the cost of the multicast tree T_1 connecting o and $m_1 \sim m_n$ directly is

$$C(T_1) = \sum_{i=1}^{n}(p_o + u_i + l_i) = np_o + \sum_i u_i + \sum_i l_i = np_o + U + L, \tag{6}$$

where $U = \sum_{i=1}^{n} u_i$ and $L = \sum_{i=1}^{n} l_i$.

In the overlay network of Fig. 5(c), the cost of the multicast tree T_2 connecting o and $m_1 \sim m_n$ passing r is

$$\begin{aligned} C(T_2) &= p_o + p_x + p_r + \sum_{i=1}^{n}(p_r + v_i + l_i) = p_o + p_x + (n+1)p_r + \sum_i v_i + \sum_i l_i \\ &= p_o + p_x + (n+1)p_r + V + L, \end{aligned} \tag{7}$$

where $V = \sum_{i=1}^{n} v_i$. Comparing Eqs (6) and (7), we can see that $C(T_2) < C(T_1)$ as long as $(n+1)p_r + p_x + (V - U) < (n-1)p_o$. If we let $\Delta = v - u$, then $(n+1)p_r < (n-1)p_o - p_x - \Delta$, i.e.,

$$p_r < \frac{(n-1)p_o - p_x - \Delta}{n+1} = \frac{n-1}{n+1}p_o - \frac{1}{n+1}(p_x + \Delta). \tag{8}$$

6.1.1. Discussions

(1) From inequation (8), $C(T_2)$ is less than $C(T_1)$ as long as p_r is little enough. At the time, r is an improving node and eligible to join the group g and *replace* o to carry out data replicating and forwarding.

(2) The right part of inequation (8) is positive when $(n-1)p_o - p_x - \Delta > 0$, i.e., $(n-1)p_o + U > p_x + V$. That is possible to improve the part of the tree if $n \geqslant 2$, and the possibility promotes as n is increasing, i.e., the children number of node o is rising.

(3) Considering the case where $p_x = 0$, i.e., $U = V$, the right part of inequation (8) must be positive if only $n \geqslant 2$. So we have

$$p_r < \frac{n-1}{n+1} p_o = \left(1 - \frac{2}{n+1}\right) p_o, \tag{9}$$

where n denotes the children number of node o which is necessary to improvement occurring. When n is large, r is qualified to become the improving node as long as p_r is a little less than p_o. We call n the *replacing hardness*. It is easy for o to find a replacing node, i.e., improving possibility is high when n is large. It is difficult when n is little, since p_r is less enough than p_o. For instance, when $n = 2$, r is o replacing node if only p_r is less than one thirds of p_o.

Especially, when $n = 1$, there is no replacing node for r and no improvement for the partial tree. On the other hand, when $n = 1$, the partial tree is equivalent to unicast path between o and m_1. Therefore, it is impossible to improve the path $o \to m_1$ when the shortest path is solving object in the case shown in Fig. 5.

(4) Inequation (9) can be rewritten as

$$\frac{p_o + p_r}{p_o - p_r} < n. \tag{10}$$

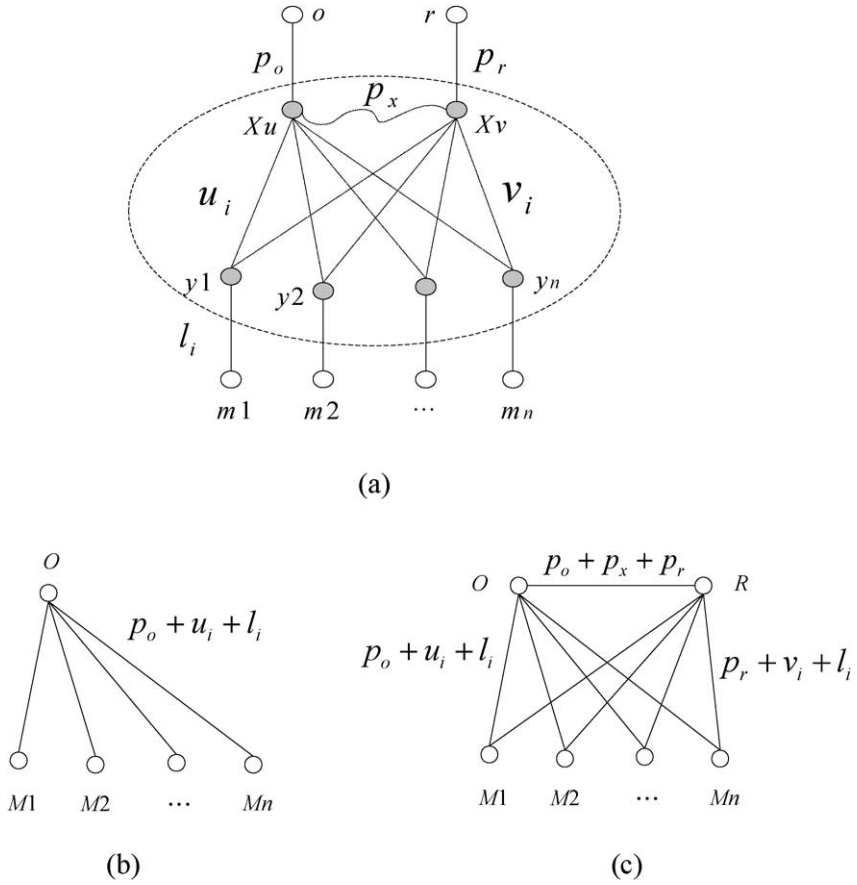

Fig. 5. A general case of tree improving.

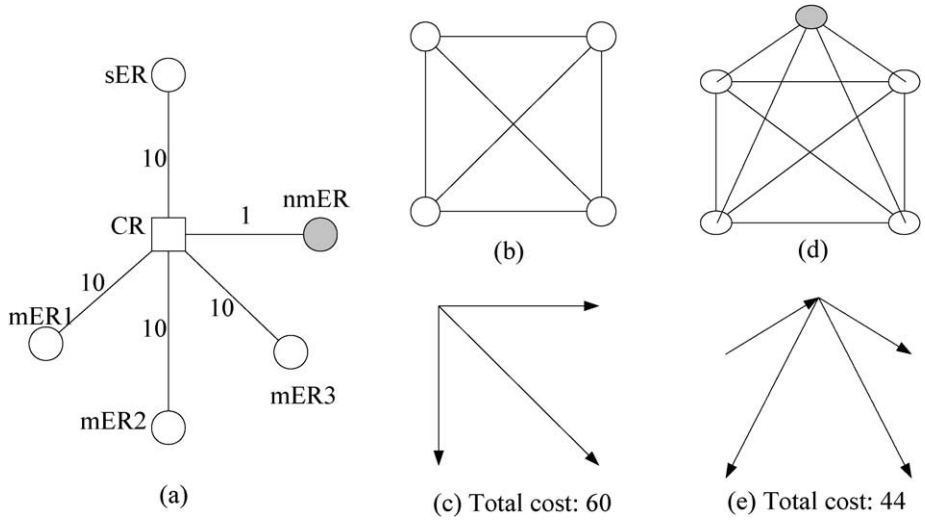

Fig. 6. An example of tree improving using a non-member ER.

Define $I_{r-o} = \frac{p_o+p_r}{p_o-p_r} = \frac{1+p_r/p_o}{1-p_r/p_o} \in [1, +\infty]$ as the *index* that r replaces o, which shows the ability of r replacing o to replicate and forward data. It only depends on p_r, the intrinsic character of the link related to r, when p_o is fixed. The less I_{r-o} (> 0), the stronger the replacing ability of r is, that is, r can replace o even if o has only a few children.

6.1.2. Example

Figure 5 shows a general case of tree improving. The following example in Fig. 6 will illustrate that the total cost of MST may be decreased if some special non-member routers are introduced into the tree.

The network domain contains five edge routers and one core router connecting each other like the star network shown in Fig. 6(a). A group session traverses the domain, involving mER1, mER2, mER3 and sER. Each member router has a link to CR with the cost of 10. And nmER is not the member of the group, from which to CR the link cost is 1. Figure 6(b) shows the overlay network among member ERs. The cost of each path between members is 20. The total cost of minimal spanning tree (MST) is 60 as shown in Fig. 6(c). But if the nmER is invited to join the group, the overlay network is shown in Fig. 6(d) and the total cost of MST is 44 as shown in Fig. 6(e), which is less than the cost of MST without nmER.

6.2. Algorithm for tree improving

Now we give an improving algorithm on how to search the improving node(s) in non-member routes for the group g, as shown in iQD-MCT (Algorithm 5).

In step (6) of the algorithm, we examine the degree limitation of current nmER checked for MRC constraint. The algorithm assumes the QoS class of improving node r is equal to the replaced node m_i by it. The exterior cycle of the algorithm runs no more than $|V_m|$, and the interior cycle runs no more than $|V_{nm}|$. Therefore, the algorithm is made to run in time $O(|V_m| \cdot |V_{nm}|) = O((|V_G|^2)/2) = O(|V_G|^2)$, for $|V_m| + |V_{nm}| = |V_G|$.

The cost of the tree after improved by Algorithm 5 is not more than that before improved, definitely as Theorem 3 shows.

Theorem 3 (*The correctness of algorithm iQD-MCT*). *On the EON being mapped into the overlay graph G, there exists a group g. If $T^P(g)$ and $T^H(g)$ are the gained trees on the subgraph $G^P(g)$ and $G^H(g)$ by Algorithm* 5*, then $C(T^H(g)) \leqslant C(T^P(g))$.*

Algorithm 5
The improving algorithm for QD-MCT, iQD-MCT

Input: Group g, tree T, member node set $V_m = \{m_i\}$, non-member node set $V_{nm} = \{nm_j\}$, node degree limitation $D^{\text{Max}}(nm_j)$, $\forall nm_j \in V_{nm}$.
Output: Improved tree T.
(0) Set the variant $R^{\text{min}} = 0$ and name current the improving node r;
(1) for each m_i in T:
(2) Set $Ch_i = \{h_k\}$, the children set of m_i on T.
(3) If $|Ch_i| \leqslant 1$, then go to (1) to check next member node;
otherwise, go to (4)
(4) Compute the cost sum P_i^m of the edges from m_i to $h_k \in Ch_i$,
$P_i^m = \sum_k C(e(m_i \rightarrow h_k))$;
(5) for each nmER_j in V_{nm}:
(6) If $D^{\text{Max}}(nm_j) < |Ch_i|$ then go to (5) to check next non-member;
(7) Compute the cost sum P_i^m of the edges from nm_j to $h_k \in Ch_i$,
$P_j^{nm} = C(e(m_i \rightarrow nm_j)) + \sum_k C(e(nm_j \rightarrow h_k))$;
(8) If $P_j^{nm} < P_i^m$ and $P_j^{nm} > R^{\text{max}}$, then $R^{\text{max}} = P_j^{nm}$, $r = nm_j$;
otherwise go to (4) to check next non-member node;
(9) If $R^{\text{min}} > 0$, then put r into T, add edges $e(m_i \rightarrow r)$ into T,
add edges $e(r \rightarrow h_k)\ \forall k$ into T and remove $e(m_i \rightarrow h_k)$;

Proof. The constituent nodes of EON are ERs. Each ER obtains the connecting information with other ERs according to its network layer routing protocols. The nodes of mER(g) related to g comprise $G^P(g)$, and $G^H(g)$ is viewed as $G^P(g)$ with one or more nmER(s). So the connecting information between mERs is unchanged. Notice that it is different from application layer multicast. Algorithm 5 only selects the nmER(s) that is able to reduce the cost of the original tree to join the tree. Consequently, $C(T^H(g)) \leqslant C(T^P(g))$. □

Assuming the replaced node set is $\{m_i\} \subseteq$ M and the replacer of m_i is n_i, the reduced cost by Algorithm 5 is $\sum_i R^{\text{min}} = \sum_i [C(e(m_i \rightarrow n_i)) + \sum_k C(e(n_i \rightarrow h_k))]$, $h_k \in Ch(m_i)$.

6.3. Difference from the SPT optimization

In *AnySee* [2], the authors improve each end user delay in the tree by non-member into the tree. Their differences from ours are twofold.

(1) The trees are built to guarantee the delay from source to every receiver is the minimum, i.e., the shortest path trees (SPTs). So the eligible node can be found by comparing the two paths delay simply. Contrastively, the proposed improving method in this paper endeavors to figure out the minimum cost trees. The searching process is more complex so as to need more topology information.

(2) The overlay nodes in AnySee are end hosts who cannot know network topology unless using additional schemes such as dispatching probe messages to other peers. However, in our scheme, the overlay nodes are edge routers who can get network topology information easily.

The above distinctions impact the improving possibility. We take an example to illustrate the possibility of improving SPT in AnySee. Assuming in Fig. 6, o, p and $m_1 \sim m_n$ are peers in AnySee. Node x_u, x_v and $y_1 \sim y_n$ are routers. On one hand, the path is the minimum from source to each receiver in the SPT, so the necessity of r being able to replace o after joining in the tree is that the path cost of $o \rightarrow r \rightarrow m_1$ is less than $o \rightarrow m_1$. That is $p_o + u_1 + l_1 > (p_o + p_x + p_r) + (p_r + v_1 + l_1)$, i.e., $u_1 > p_x + 2p_r + v_1$. We get $u_1 > p_x + v_1$ for $p_r > 0$. It sounds that improving possibility is promoted. On the other hand, however, if routing metric in router x_u and x_v is identical to the cost of paths, then x_u will know path $x_u \rightarrow x_v \rightarrow y_1$ is more optimal than path $x_u \rightarrow y_1$ and prefer the former routing. This is the case mentioned in the Discussion (3) (Section 6.1.1). As a result, no improvement

exists. Only if the routing metric is different from the cost of paths, x_u will not know $x_u \to x_v \to y_1$ is the optimal path. At the moment, it is possible for upper layer applications to find improving nodes. However, the proposed method for the total tree cost always has the opportunity to find improvement.

7. Simulation experiments

In this section, we evaluate the proposed tree-building algorithms through simulations. We first give the performance measurements used in simulation experiments.

7.1. Performance measurements

(1) *The tree performance*

Computing the optimal solution, T^{opt} to the QD-MCT problem is difficult since it is NP hard. We replace T^{opt} with T^{Q}, which is the multi-class QoS minimum cost tree without degree limitation, i.e., the optimal solution of the Q-MCT problem in the pure subgraph. Obviously, $C(T^{Q}) \leqslant C(T^{\text{opt}})$. According to Theorem 2, the optimal solution can be gained using the Q-MCT algorithm (Algorithm 1). The solutions of the Algorithms 2–4 are denoted as T_{An}^{QD}, $n = 2, 3$ and 4. Let λ_{An} ($n = 2, 3, 4$) denote the performance ratios for Algorithms 2–4 as follows.

$$\lambda_{An} = \frac{C(T_{An}^{QD}) - C(T^{Q})}{C(T^{Q})} \times 100\%, \quad n = 2, 3, 4. \tag{11}$$

The smaller λ_{An} means better performance.

(2) *For the tree improvement*

To validate the performance of the tree improving algorithm (Algorithm 5), we use the algorithm to improve the trees computed in advance by Algorithms 1 and 2. The improving performance of trees computed by Algorithms 3 and 4 are like the one of Algorithm 2. The improved tree of Algorithm 1 is called T^{iQ}; the improved tree of Algorithm 2 is called T^{iQD} if no misleading. We let the μ_1 and μ_2 denote the improving effects for the trees with and without constraints, respectively, as shown as follows.

$$\mu_1 = \frac{C(T^{Q}) - C(T^{iQ})}{C(T^{Q})} \times 100\%, \qquad \mu_2 = \frac{C(T^{QD}) - C(T^{iQD})}{C(T^{QD})} \times 100\%. \tag{12}$$

Also, we let the λ_{A5} denote the performance ratios of the improving algorithm as below,

$$\lambda_{A5} = \frac{C(T^{iQD}) - C(T^{iQ})}{C(T^{iQ})} \times 100\%. \tag{13}$$

7.2. Setting of experiments

We generate the experiment topologies using GT-ITM. Each node denotes a router and the total number of nodes is N. ER nodes are a half of the nodes. Random variant $\beta \in (0, 1)$ is the rate of mER number to ER number. The group size is $N_{\text{mER}} = \beta N/2$. The link cost between nodes is refered to as the corresponding edges delay, designated by the topology generator. The edge cost between two ERs is the sum cost of all the links on the shortest path connecting them. The degree limitation for each node is $D^{\text{Max}} \geqslant 2$ and QoS class number is $Q^{N} = 3$ in simulations.

Two types of topology graph are generated: the sparse mode and the dense mode. The former holds more edges than the latter with the same number of nodes. It is implemented by setting the density parameter $p \in (0, 1)$ which is the possibility of existing an edge between any two nodes in the graph.

To evaluate performance of the proposed algorithms, we set $(N, p) = \{(1000, 0.8), (1000, 0.2), (500, 0.8), (500, 0.2)\}$ in the simulations, which denote the topology with many or few nodes and dense or sparse edges, respectively. In each group experiment, we set $\beta = \{0.05, 0.1, 0.2, \ldots, 0.9\}$, totaling ten, and set $D^{\text{Max}} = 10$ for the case of equal degree limitations and random $D^{\text{Max}} \in [2, 9]$ for the case of inequal degree limitations. We set $k = 5$ for the DH algorithm and $\alpha = 0.8$ for the CH algorithm.

To examine the impact of k on the DH algorithm, we set $\beta = 0.5$ constantly and set $k = \{1, 2, \ldots, 10\}$ for evaluating the algorithm performance. To examine the impact of α on the CH algorithm, we set $\beta = 0.5$ constantly and set $\alpha = \{0, 0.1, 0.2, \ldots, 0.9\}$ for evaluating the algorithm performance.

In these simulations, a few of links between ERs are set randomly to be dissatisfactory for the required QoS class. The link number is less than 5%.

7.3. Experiment results

(1) *Performance ratios*

The performance ratios of the QD-Heuristic, DH and CH algorithms are shown in Fig. 7 where node number in the overlay network is 250. Figure 7(a) and (b) shows the dense mode, i.e., the case of more edges ($p = 0.8$); Fig. 7(c) and (d) shows the sparse mode, i.e., the case of less edges ($p = 0.2$). And Fig. 7(a) and (c) shows the case of equal degree limitations, with $D^{\text{Max}} = 10$; Fig. 7(b) and (d) shows the case of unequal degree limitations with D^{Max} in 2–9 randomly. In each figure, the x-coordinate denotes the β, i.e., the rate of mER number to ER number, and the y-coordinate denotes the performance ratios.

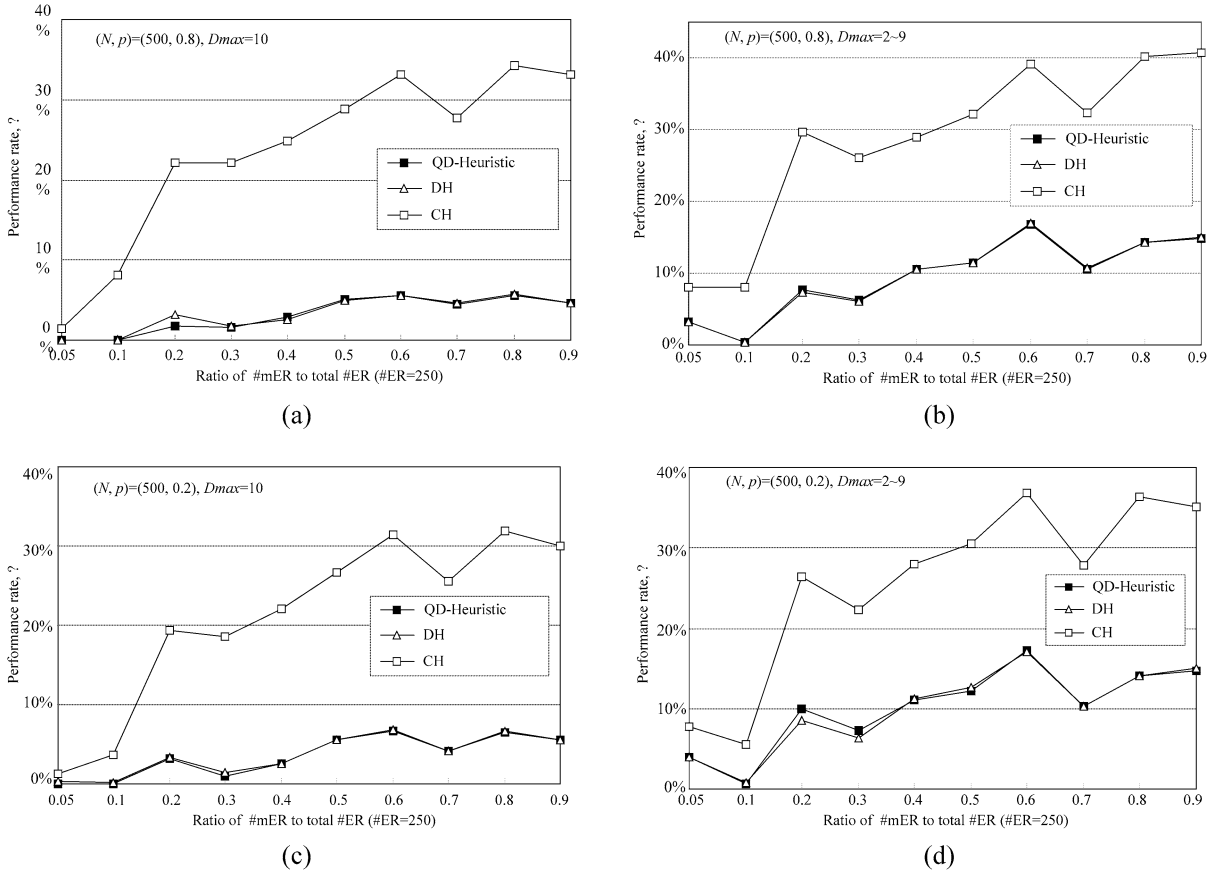

Fig. 7. The case of 250 nodes in the overlay network.

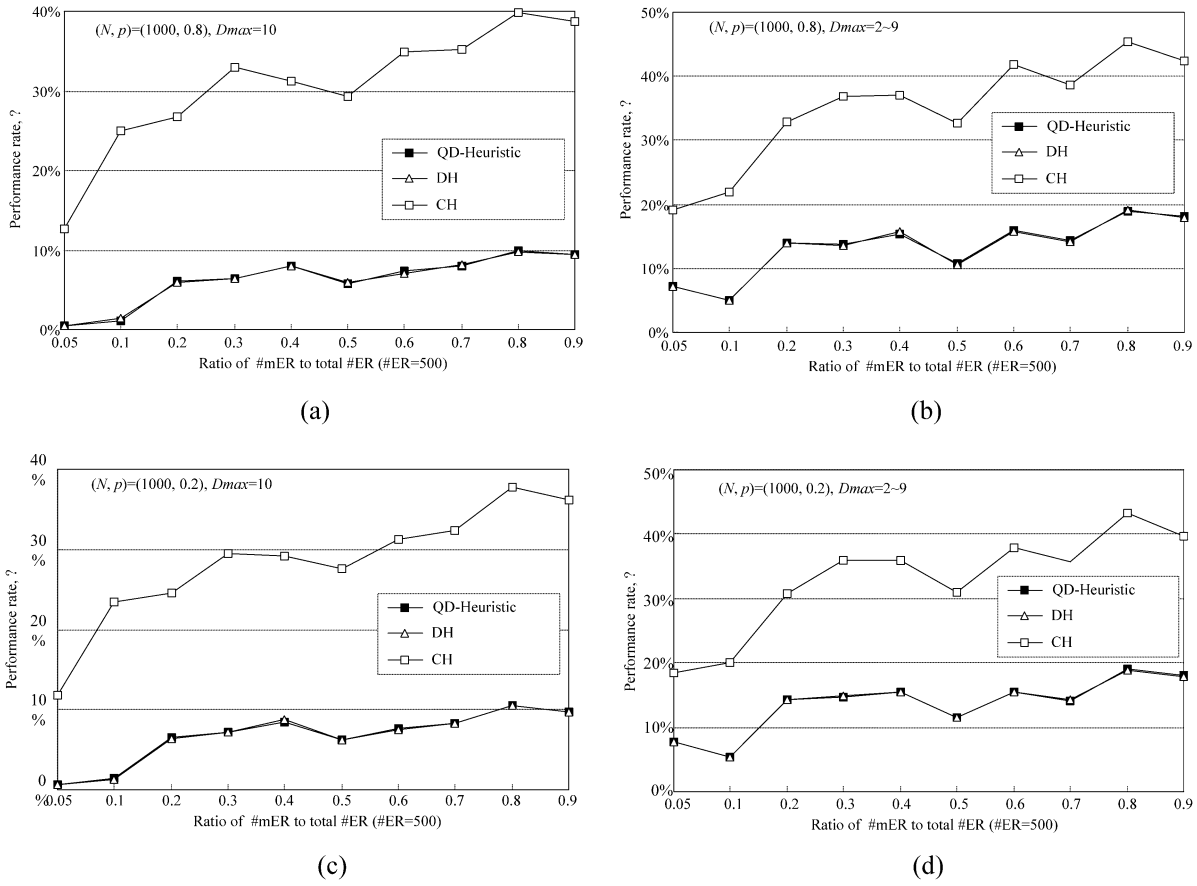

Fig. 8. The case of 500 nodes in the overlay network.

From Fig. 7, we can see that the performance of QD-Heuristic is better than the CH algorithm. The solved tree cost of the QD-Heuristic exceeds the result of Q-MCT algorithm at most 6% in the case of equal degree limitations and at most 17% in the case of unequal degree limitations. It shows the advantage of the QD-Heuristic. The DH algorithm is close to the QD-Heuristic at performance ratios. Under some conditions, the result of the DH algorithm is better than the one of the QD-Heuristic, which demonstrates the effectiveness of the former.

We also find from the curves in Fig. 7 that the results of these three algorithms in the case of unequal degree limitations are all worse than the ones in the case of equal degree limitations. The reason is that the average of degree limitations in the case of unequal degree is smaller than 10 for the case of equal degree. So chance of finding smaller cost tree is reduced in the case of unequal degree.

The performance ratios of the proposed algorithms are shown in Fig. 8 where node number in the overlay network is 500. The analysis results are similar to Fig. 7. At the same time, we can see that the performance of these algorithms decreases as node number increasing in the overlay network QD-Heuristic is the best in these algorithms. The λ of the QD-Heuristic is lower than 20% in the figure, which is 11% at most in the case of equal degree limitations and 19% at most in the case of unequal degree limitations.

(2) *Running time and iteration times*

As node number of multicast member increasing, the running time of these four algorithms is shown in Fig. 9 where node number of the overlay network is 250 and probability of link existing $p = 0.2$. The results are the average simulation running time of 100 times. Figure 9(a) and (b) shows the cases of equal degree limitations and

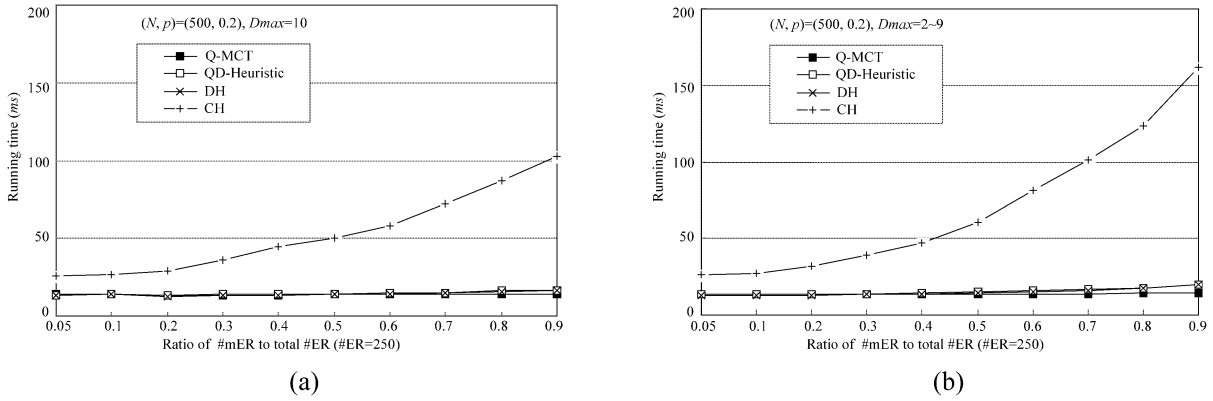

Fig. 9. Running time of four algorithms.

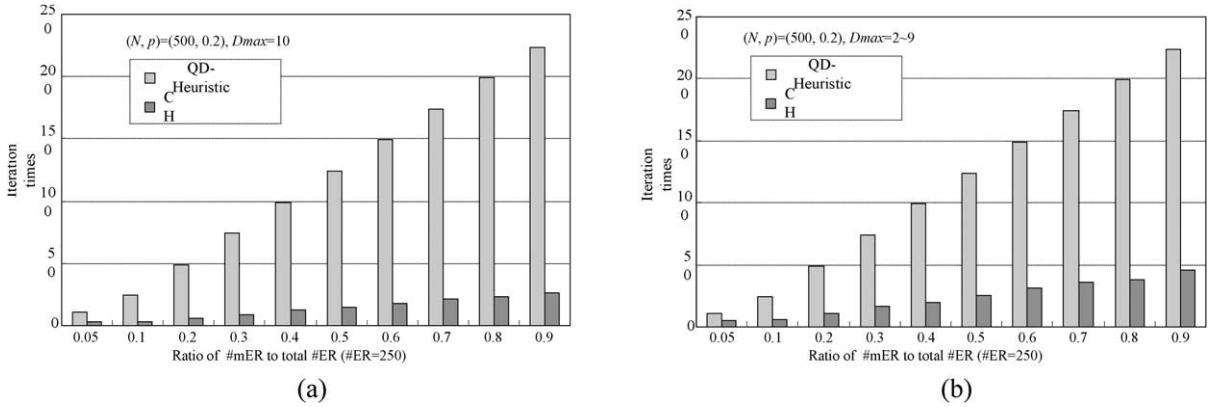

Fig. 10. Iteration times in CH and QD-Heuristic algorithms.

unequal degree limitations respectively. The x-coordinate denotes the β; the y-coordinate denotes the running time in the unit of millisecond (ms). From this figure, the running time of the Q-MCT algorithm is the shortest, the QD-Heuristic and the DH algorithm are close to the Q-MCT algorithm, and the running time of CH algorithm is the longest.

Although running time of the CH algorithm is longer, its iteration times are smaller as shown in Fig. 10 where the node number is 250 in the overlay network and $p = 0.2$. Figure 10(a) and (b) shows the cases of equal and unequal degree limitations. The iteration times for building the tree in the CH algorithm increase more slowly than in the QD-Heuristic, as member node number is increasing.

(3) *The impact of k in the DH algorithm*

In each iteration cycle of the DH algorithm, k off-tree nodes being closest to the tree are selected to compose candidate node set K where the node of the largest residual degree is elected to join the tree. Figure 11 shows the impact of k on the algorithm performance. The x-coordinate denotes the ten values of k from 1 to 10, and the y-coordinate denotes the difference of the performance ratios of the DH algorithm and the QD-Heuristic, i.e., $\Delta\lambda = \lambda_{DH} - \lambda_{QD\text{-}Heuristic}$. The eight curves in the figure indicate the $\Delta\lambda$ changing under various node numbers, various probability of link existing and various degree limitations. The DH algorithm becomes the QD-Heuristic when $k = 1$. From the Fig. 11, we can see that the $\Delta\lambda$ of each case is less than 0 except the case of $(N, p, D^{\text{Max}}) = (500, 0.2, 2\sim9)$ when $k \geqslant 4$. That is to say, the tree of DH algorithm is less than that of QD-Heuristic. The $\Delta\lambda$ of DH algorithm decreases as k is increasing.

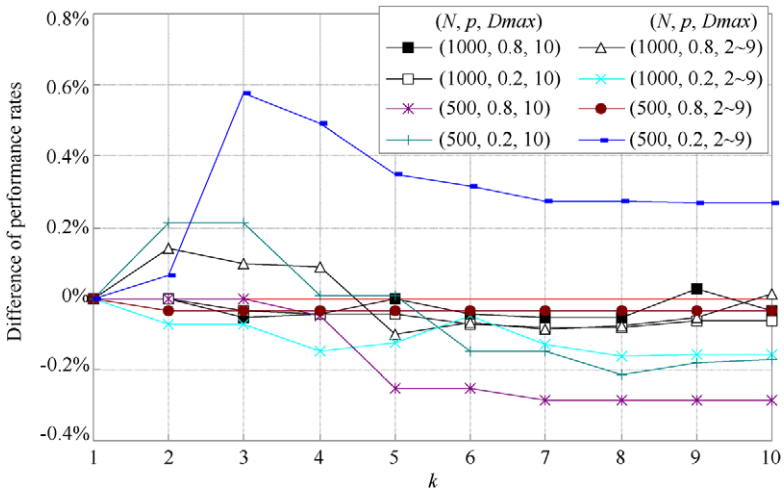

Fig. 11. Impact of k on DH algorithm.

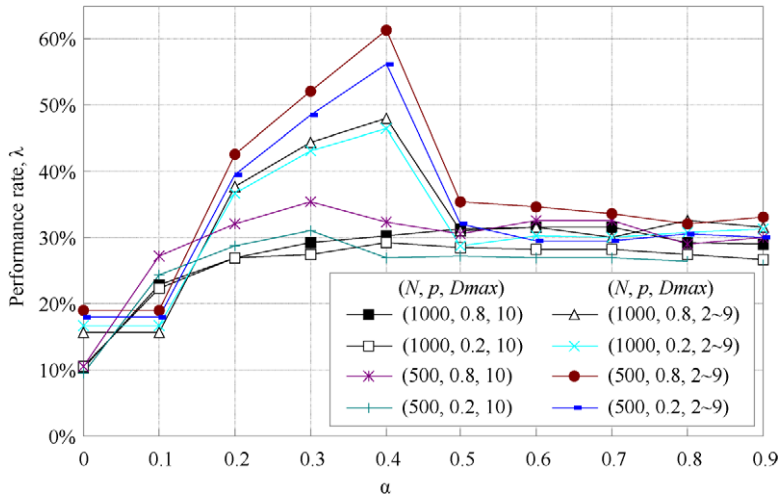

Fig. 12. Impact of α on CH algorithm.

(4) *The impact of α in the CH algorithm*

In the CH algorithm, the $Clu(u)$ size must be less than $D^{\text{Max}}(u)$. We set $Nc(u) = \lceil \alpha D^{\text{Max}}(u) \rceil, 0 < \alpha < 1$. Figure 12 shows the impact of the various α in the algorithm. The x-coordinate denotes the ten values of α from 0 to 0.9, and the y-coordinate denotes the performance ratios of λ_{CH}. The eight curves in the figure indicate the λ_{CH} changing under various node numbers, various probability of link existing and degree limitations. From Fig. 12, we can see that λ_{CH} climbs to the peak value at $\alpha = 0.3$ when node degree limitations are equal to each other, and λ_{CH} is decreasing as α increasing. When node degree limitations are unequal to each other, λ_{CH} climbs to the peak value at $\alpha = 0.4$ when node degree limitations are unequal to each other, and decreases as α increasing. When $\alpha \geqslant 0.5$, the results of the algorithm are stable.

(5) *Tree improving performance*

The improving effects of iQD-MCT are shown in Fig. 13. It consists of four group experiments, setting $(N, p) = \{(1000, 0.8), (1000, 0.2), (500, 0.8), (500, 0.2)\}$, which denote the topology with many or few nodes and dense or

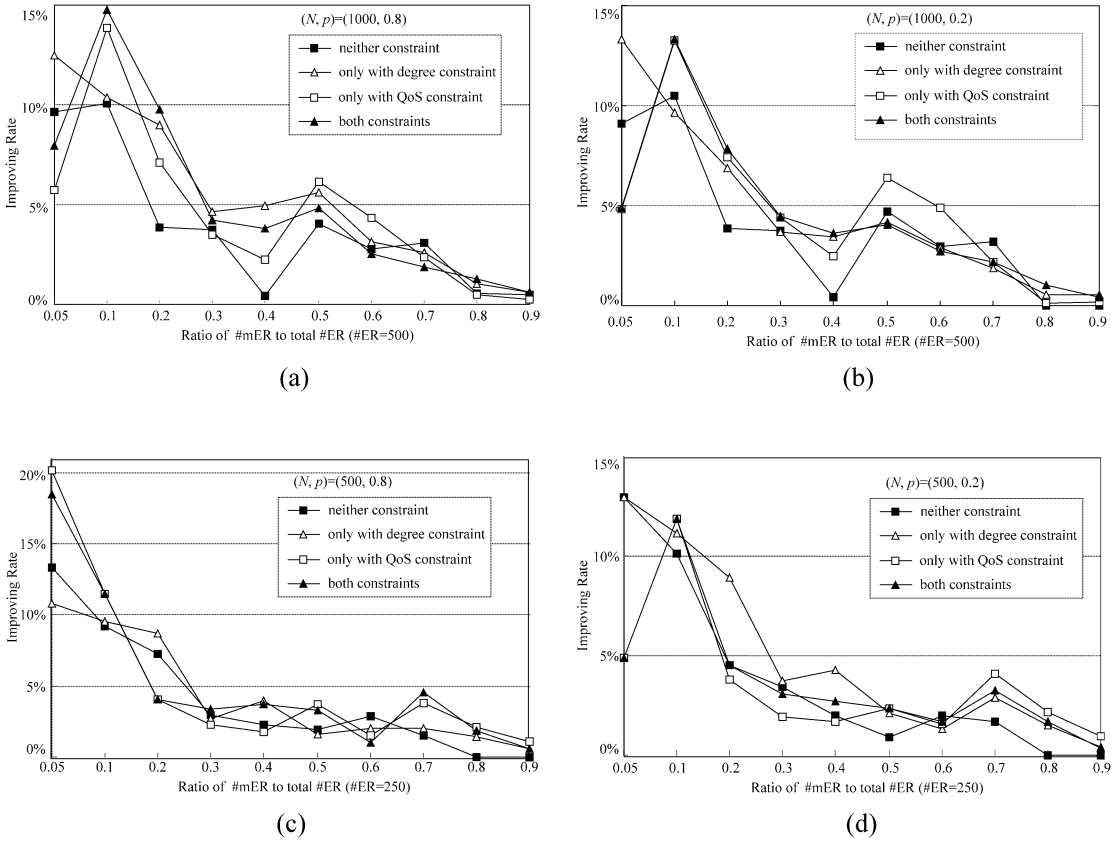

Fig. 13. Tree improving performance (I).

sparse edges respectively. In each group experiment, $\beta = \{0.05, 0.1, 0.2, \ldots, 0.9\}$, totaling ten. For the cases with or without MQC and MRC, the $(Q^N, D^{\text{Max}}) = \{(0,0),(0,10),(3,0),(3,10)\}$.

According to the results of Fig. 13(c), the values of μ_1 and μ_2 are both about 5%. The summit of improving effect in QD-MCT is approaching 18% when $\beta = 0.05$. Moreover, we can obtain two facts from these curves. One is that the less β, the better the improvement is. The maximum effects occur at $\beta = 0.1$ in the four group curves, while there is nearly no improvement when $\beta > 0.8$. The reason is that the less value of β means the larger number of nmERs, so there is more likelihood of finding eligible nmERs to improve the tree. The other is that improvement to the trees with constraints is larger than the one without any constraint. Under the constraint conditions, the performance of the trees by Algorithm 2 is worse than the trees without constraints. Therefore, they have more opportunity to be improved.

Figure 14 shows the algorithms performance changing as group size N_{mER} increasing in the case of $(\beta, p) = (0.4, 0.5)$. Total eight values of N_{mER} are examined: 25, 50, 75, 100, 125, 150, 175 and 200. In the figure, the top part of curves displays the cost of the trees found and improved by the two algorithms, respectively. The medium part of columns displays the improving effect by Algorithm 2, and the bottom part shows the performance rate of Algorithm 1. The figure reveals similar results with the above analysis.

8. Conclusions

This paper presents an overlay multicast scheme supporting comparable multi-class QoS in the general network domains. In the scheme, the overlay network is constructed on edge routers of the do-

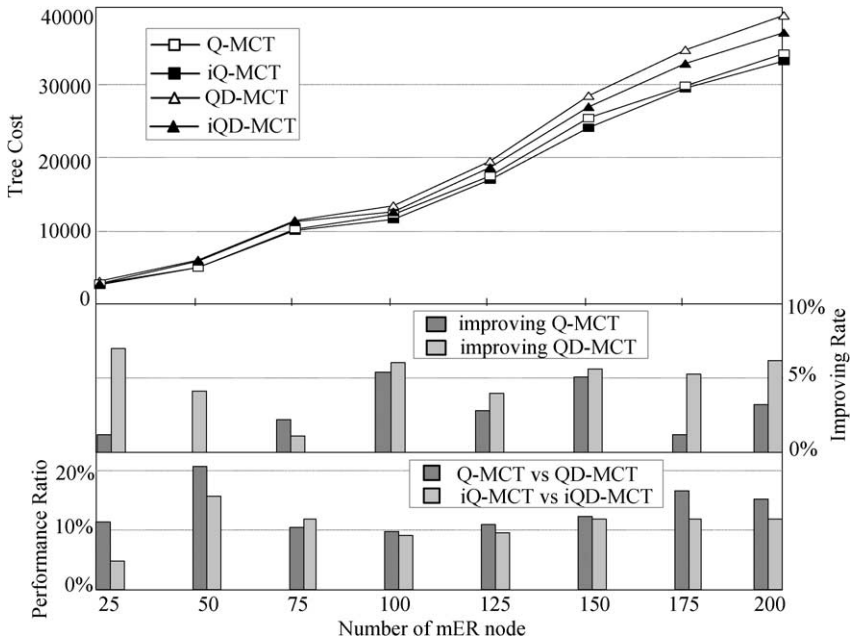

Fig. 14. Tree improving performance (II).

main, and on the overlay network multicast is implemented. The minimum cost trees are built to delivery traffic so as to increase total network performance. The scheme supports different QoS requirements by members, and considers the member routers' power for replicating and forwarding in overlay multicast trees. In order to achieve these targets, the multicast route problem is introduced to satisfy the multi-class QoS and node degree limitations. Due to hardness of the problem, three tree building heuristics are proposed.

The QD-Heuristic algorithm has a good performance with the approximation ratio of $C_{\max}/C_{\min}$, where $C_{\max}$ and $C_{\min}$ are the maximum and minimum cost of all the edges in the overlay network respectively; it has a time complexity of $O(n + m + n \log n)$, where n is member node number and m is edge number. The node residual degree based DH algorithm is a revision of the QD-Heuristic, whose time complexity is $O(n + m + kn \log n)$. The cluster based CH algorithm has a time complexity of $O(m + Nc \cdot n \log n + \theta n \log \theta n + \theta n^2)$, where *Nc* is the cluster size and θ is the ratio of cluster number and node number. The simulation results demonstrate that the QD-Heuristic performance is the best and the solution of the DH algorithm is greatly close to the QD-Heuristic solution. These two algorithms are better than the CH algorithm in performance and running time. On the other hand, the CH algorithm needs smaller iteration times to build trees than the former two algorithms.

The proposed scheme offers an option to implement QoS multicast in DiffServ domains or MPLS VPN networks. It can compensate some disadvantages of the multicast VPNs specified in the draft [11].

8.1. Future work

We overlook the fact in this paper that edges in the overlay network are not independent of each other, since the core path they represent may have core edges in common. Thus the cost of the core edge may be affected by multicast traffic and the fact that multiple multicast trees traverse the same core edge. This is an issue that has to be addressed in future work. We would like to study the deployment of the multicast scheme across multiple network domains. It is also interesting in how to solve the minimum delay tree or other types of the trees with these two constraints mentioned.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant Nos 60303006 and 60473082, the National Grand Fundamental Research 973 Program of China under Grant No. 2003CB314801 and the National High-Tech Research and Development 863 Plan of China under Grant No. 2006AA01Z209. Thanks to the reviewers of the *Journal of High Speed Networks*.

References

[1] Y. Chu, S.G. Rao and H. Zhang, A case for end system multicast, in: *Proceedings of ACM Sigmetrics*, June 2000.

[2] X. Liao, H. Jin, Y. Liu, L. Ni and D. Deng, AnySee: Peer-to-peer live streaming, in: *Proceedings of IEEE INFOCOM*, April 2006.

[3] M. Kwon and S. Fahmy, Topology-aware overlay networks for group communication, in: *Proceedings of ACM NOSSDAV*, May 2002.

[4] S. Shi and J. Turner, Multicast routing and bandwidth dimensioning in overlay networks, *IEEE Journals on Selected Areas in Communications (JSAC)* **20**(8) (2002), 1444–1455.

[5] A. Striegel, A. Bouabdallah, H. Bettahar and G. Manimaran, EBM: A new approach for scalable DiffServ multicasting, in: *Proceedings of International Workshop on Networked Group Communications (NGC)*, 2003.

[6] M. Gupta and M. Ammar, Providing multicast communication in a differentiated services network using limited branching techniques, in: *Proceedings of International Conference on Internet Computing (IC)*, June 2002.

[7] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, An architecture for differentiated services, IETF RFC 2475, 1998.

[8] A. Fei, J. Cui, M. Gerla and M. Faloutsos, Aggregated multicast: An approach to reduce multicast state, in: *Proceedings of Sixth Global Internet Symposium (GI2001)*, November 2001.

[9] S. Li, J. Wu, K. Xu and Y. Liu, A modularized QoS multicasting approach on common homogeneous trees for heterogeneous members in DiffServ, in: *Proceedings of IEEE International Performance Computing and Communications Conference (IPCCC)*, April 2006.

[10] E. Rosen and Y. Rekhter, BGP/MPLS IP Virtual Private Networks (VPNs), IETF RFC 4364, 2006.

[11] E. Rosen and R. Aggarwal, Multicast in MPLS/BGP IP VPNs, Internet draft: draft-ietf-l3vpn-2547bis-mcast-01.txt, December 2005.

[12] T. Cormen, C. Leiserson and R. Rivest, *Introduction to Algorithms*, McGraw Hill, 2000.

[13] C. Oliveira and P. Pardalos, A survey of combinatorial optimization problems in multicast routing, *Computers and Operations Research* **32**(8) (2005), 1953–1981.

[14] B. Wang and J. Hou, Multicast routing and its QoS extension: Problems, algorithms and protocols, *IEEE Network* **14**(1) (2000), 22–36.

[15] A. Striegel and G. Manimaran, A survey of QoS multicasting issues, *IEEE Communications Magazine* **40**(6) (2002), 82–87.

[16] M. Faloutsos, A. Banerjea and R. Pankaj, QoSMIC: Quality of service sensitive Multicast Internet protoCol, in: *Proceedings of ACM SIGCOMM'98*, Vancouver, British Columbia, September 1998.

[17] S. Chen, K. Nahrstedt and Y. Shavitt, A qos-aware multicast routing protocol, in: *Proceedings of IEEE INFOCOM*, March 2000.

[18] A. Fei and M. Gerla, Receiver-initiated multicasting with multiple QoS constraints, in: *Proceedings of IEEE INFOCOM*, March 2000.

[19] J. Hou, H. Tyan, B. Wang and Y. Chen, QoS extension to CBT, Internet draft: draft-hou-cbt-qos-00.txt, February 1999.

[20] S. Biswas, R. Izmailov and B. Rajagopalan, A QoS-aware routing framework for PIM-SM based IP-multicast, Internet draft: draft-biswas-pim-sm-qos-00.txt, June 1999.

[21] S. McCanne, V. Jacobson and M. Vetterli, Receiver-driven layered multicast, in: *Proceedings of ACM SIGCOMM*, New York, 1996.

[22] S. Cheung, M. Ammar and X. Li, On the use of destination set grouping to improve fairness in multicast video distribution, in: *Proceedings of IEEE INFOCOM*, 1996, pp. 553–560.

[23] D. Yang and W. Liao, MQ: an integrated mechanism for multimedia multicasting, *IEEE Transaction on Multimedia* **3**(1) (2001), 82–97.

[24] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin, Resource ReSerVation Protocol (RSVP), IETF RFC 2205, 1997.

[25] R. Braden, D. Clark and S. Shenker, Integrated services in the Internet architecture: an overview, IETF RFC 1633, 1994.

[26] R. Bless and K. Wehrle, A lower than best-effort per-hop behavior, Internet draft: draft-bless-diffserv-lbe-phb-00.txt, September 1999.

[27] Z. Li and P. Mohapatra, QoS-aware multicasting in DiffServ domains, in: *Proceedings of IEEE Global Internet Symposium*, November 2002.

[28] G. Bianchi, N. Blefari-Melazzi, G. Bonafede and E. Tintinelli, QUASIMODO: Quality of service-aware multicasting over Diffserv and overlay networks, *IEEE Network* **17** (2003), 38–45.

[29] B. Yang and P. Mohapatra, Multicasting in differentiated service domains, in: *Proceedings of IEEE GLOBECOM*, November 2002.

[30] A. Striegel and G. Manimaran, A scalable approach for DiffServ multicasting, in: *Proceedings of IEEE ICC*, Helsinki, Finland, June 2001.

[31] R. Boivie, N. Feldman, Y. Imai, W. Livens, D. Ooms and O. Paridaens, Explicit Multicast (Xcast) Basic Specification, Internet draft: draft-ooms-xcast-basic-spec-01.txt, March 2001.

[32] N. Wang and G. Pavlou, An overlay framework for provisioning differentiated services in Source Specific Multicast, *Computer Networks* **44**(4) (2004), 481–497.

[33] S. Bhattacharyya, An Overview of Source Specific Multicast (SSM), IETF RFC 3569, 2003.

[34] A. Sudhir, G. Manimaran and P. Mohapatra, Heterogeneous QoS multicast in DiffServ-like networks, in: *Proceedings of IEEE ICCCN*, October 2005.

[35] J. Cui, L. Lao, M. Faloutsos and M. Gerla, AQoSM: Scalable QoS multicast provisioning in Diff-Serv networks, *Computer Networks* **50** (2006), 80–105.

[36] M. Bishop, S. Rao and K. Sripanidkulchai, Considering priority in overlay multicast protocols under heterogeneous environments, in: *Proceedings of IEEE Infocom*, April 2006.

[37] J. Liang and K. Nahrstedt, RandPeer: Membership management for QoS sensitive peer-to-peer applications, in: *Proceedings of IEEE Infocom*, April 2006.

[38] L. Subramanian, I. Stoica, H. Balakrishnan and R. Katz, OverQoS: An overlay based architecture for enhancing internet QoS, in: *Proceedings of 1st Symposium on Networked Systems Design and Implementation (NSDI)*, March 2004.

[39] M. Garey and D. Johnson, *Computers and Intractability, a Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., 1979.