结合资源预留的分布式 QoS 组播路由算法

杜 荔 刘玉涛
(东北大学信息科学与工程学院 沈阳 110004)

摘 要：针对网络资源信息的动态变化对 QoS 组播路由算法的巨大影响，该文提出了一种与资源预留结合的分布式组播路由算法 DQMTR。DQMTR 通过在路径探索过程中进行资源预留克服网络信息变化对 QoS 路由算法的影响。DQMTR 还通过记录预约资源的数量解决资源的过预约问题，并利用 DiffServ 体系下 QoS 路由算法的特点使算法能够适用于 DiffServ 网络。仿真实验表明，DQMTR 提高了组成员加入的成功率，优化了平均路径代价值。

关键词：组播；区分服务；资源预留；非确定环境
中图分类号：TP393 **文献标识码**：A **文章编号**：1009-5896(2009)01-0210-05

The Algorithm for Distributed QoS Multicast Routing with Resource Reservation

Du Li Liu Yu-tao
(*College of Information Science & Engineering*, *Northeastern University*, *Shenyang* 110004, *China*)

Abstract: To eliminate the affect of the change of resource information on the QoS multicast routing, an algorithm for Distributed QoS Multicast Routing with Resource Reservation(DQMTR) is put forward in this paper. Using resource reservation in the process of path-detecting, DQMTR overcomes the issue. By noting the number of reserved resource, DQMTR resolves the issue of over-reservation. This paper considers the characteristic of differentiated services model to which DQMTR is adapted. The simulation reveals that the algorithm improves the success ratio, and optimizes the average path cost.

Key words: Multicast; Differentiated services; Resource reservation; Uncertain condition

1 引言

QoS(Quality of Service)组播路由问题是目前网络领域中的热点问题之一，传统对 QoS 组播路由问题的研究是在设定相关的网络信息为确定的前提下而展开的，而实际网络环境中的网络信息是动态变化的，而且由于大型网络的分层机制、信息的隐藏、信息计算的不精确等因素[1]造成了结点所获得的网络全局状态并非是确定的。本文只讨论网络资源数量的动态变化所造成的信息的非精确性问题，这种非精确性会对域内路由算法的性能产生严重影响，因此也成为 QoS 路由研究的主要难点之一[2]。

对于非确定环境下的 QoS 路由问题，目前已有的算法大多是根据链路状态信息的概率分布情况计算出最有可能满足约束条件的路径[3−6]，为了利用各条链路状态信息的概率分布函数，需要将端到端 QoS 需求分配到连接的各条链路上，再根据分配的结果来预约相应链路上相应数量的资源，以期达到较大的路径建立的成功率。算法大多假定网络资源数量服从一种概率分布模型(如均匀分布)，这种假定多数情况下不符合实际情况。如果根据真实网络中资源数量变化的统计特性来建立并维护合适的概率分布函数，则必然会给网络造成很大的负担。这些算法都是集中式路由算法或应用于集中式路由算法，且以集成服务(IntServ)模型为背景。集中式路由算法由于其自身的特性决定了它不适用于较大规模的网络，其应用范围比较小。况且，IntServ 也不适用于较大规模的网络[7]。

分布式路由算法由于可扩展性好、节点负担比较均匀等特点受到了广泛的关注。它与集中式路由算法相比有很多优越性，使得分布式路由算法有着广泛的发展空间。而且，由于分布式路由算法在寻路过程中信令的逐跳转发策略，使得信令能及时捕捉资源信息的变化，从而受网络信息的非精确性的影响也比较小。在与资源预留结合的情况下，算法可以克服这种非精确性的影响[8]。

本文以区分服务(DiffServ)体系作为 QoS 路由算法的应用背景，分析 DiffServ 体系下 QoS 路由算法的特点，提出基于 ticket 的 QoS 组播路由算法，分析该算法的时间复杂度，根据实验结果探讨该算法与资源预留结合时的性能。

2 DiffServ 体系下 QoS 组播路由算法特点

2.1 QoS 组播路由问题描述

设网络 $G=<V, E, H>$为一有向连通赋权图，其中 V 为网络中所有交换节点的集合，E 为任意两节点间通信链路

2007-07-03 收到，2007-12-14 改回
国家 863 计划项目(2002AA784030) 资助课题

集合，H 为链路上度量函数的集合，链路的资源信息就是由度量函数来描述的。若 $e=(s, t)\in E$，s，$t\in V$，因为网络链路实际上是不对称的，则必存在 $e'=(t, s)\in E$，即表示 G 为双向全工网络。对任意链路 $e\in E$，$H=\{\text{Bandwidth}(e), \text{Delay}(e), \text{Lost}(e), \text{Cost}(e)\}$。其中，带宽函数 $\text{Bandwidth}(e)$，延迟函数 $\text{Delay}(e)$，丢失率函数 $\text{Lost}(e)$，代价函数 $\text{Cost}(e)$ 分别表示链路 e 的可用带宽、延迟、丢失率和代价。

组播树 $T=<s, D_T, V_T, E_T>$，其中 $V_T \subseteq V$，$E_T \subseteq E$ 分别代表组播树上的路由器节点集合和链路集合，$s \in V_T$ 为数据发送源主机的默认路由器节点，D_T 为组成员主机的默认路由器节点集合，本文称 D_T 中的元素为组成员。对于任意一个 $d \in D_T$，$P_T(s, d)$表示 s 与 d 之间沿着 T 的路径。设 d 的带宽约束要求为 B_d，延迟约束要求为 D_d，丢失率约束要求为 L_d，则有式(1)-式(3)成立。

$$\min\{e \mid e \in P_T(s,d)\} \geq B_d \tag{1}$$

$$\sum_{e\in P_T(s,d)} \text{Delay}(e) \leq D_d \tag{2}$$

$$1-\prod_{e\in P_T(s,d)} (1-\text{Lost}(e)) \leq L_d \tag{3}$$

即对于 D_T 中每一个成员，上述 3 式都必须同时满足，并且使式(4)的值最小化。

$$\sum_{e\in P_T(s,d)} \text{Cost}(e) \tag{4}$$

QoS 组播路由算法的作用就是为每一个请求加入组播树的节点选择出满足式(1)-式(3)并使式(4)最小的路径，最终形成一颗组播树。

2.2 DiffServ 体系下 QoS 组播路由模型

DiffServ 模型把复杂的流量调节功能和每个数据流的状态信息推向网络的边缘，内部节点只处理聚合后的数据流，而非单个流。DiffServ 根据由 DSCP 标识的数据流的不同等级提供不同质量的服务，同一等级的所有数据流会得到相同的服务。以延迟约束为例，DiffServ 对同一等级的所有数据流提供的延迟约束保证是相同的。故当 DiffServ 经过协商确定一个请求的服务质量等级之后，就可以知道相应的数据流在传送过程中网络各个节点所能保证的延迟约束上限 $\text{Delay}(e)$，且在一个 DiffServ 域内这个上限对于所有节点都是相同的，记为 Delay。对丢失率的讨论也是类似的，记单个节点丢失率上限为 Lost。设 $P_T(s, d)$的长度为 l，则式(2)、式(3)变为

$$l \times \text{Delay} \leq D_d \tag{5}$$

$$1-(1-\text{Lost})^l \leq L_d \tag{6}$$

在所有同时满足式(5)，式(6)的正整数 l 中选择最大的作为跳数约束，记作 h。则 $P_T(s, d)$的长度 l 必须满足：

$$l \leq h \tag{7}$$

这样，在 DiffServ 下，组成员对延迟和丢失率的约束要求转变为对路径长度即跳数的约束。即 QoS 组播路由算法的作用就是为每一个请求加入组播树的节点选择出满足式(1)和式(7)并使式(4)的值最小的路径。

3　DQMTR 算法描述与时间复杂度分析

3.1 DQMTR 算法基本原理

本算法限定组播树建立过程中各组成员的加入是按照一定的次序完成的。即最初组播树中只有源节点，然后组成员一个一个地与源节点建立单播路径。第 1 个组加入申请的路由计算过程与单播路由无异，但以后组成员加入过程中的路由计算要受已有组播树的影响。当一个成员发起的路由过程完全结束后，下一个成员才能申请加入组播树。

算法的基本思想是，对于每一个组加入请求，源节点发送一定数量的探测分组，这个数量由 ticket 的值控制，这样分组将沿着不同的路径到达目的节点。在分组转发过程中每到一个节点便进行一次接纳控制，若接纳控制成功则软资源预约，然后前进到下一跳节点，否则分组被传回上游节点重新进行选路，并且在传送过程中分组记录沿途经过的路径和代价。这样到达目的节点的所有分组所包含的路径都是可行路径。分组到达目的节点时，目的节点会在所有探测分组所包含的路径中选择一条代价最小的路径建立连接。其它分组按照原路返回，并释放预约的资源。

资源预留的目的是使节点能够实现对数据流的 QoS 保证。网络系统资源大致包含 3 个部分：缓冲区、链路带宽和处理器资源。QoS 路由过程中预留的资源是指链路带宽和缓冲区资源。为了提高资源利用率，本文采用“软预约”[9]作为资源预留的方式。被软预约的资源，不能被其它业务预约，但可以被使用。一旦被软预约的资源转变为“硬”预约，正在使用这些资源的业务必须让出它们。

在结合资源预留的分布式组播路由算法进行路由时会引起资源的“过预约”问题，即重复预约了多份资源，造成了资源不必要的浪费。本算法对这一问题的解决方法是，分组在第 1 次预约资源的时候需记录预约的数量，这个数值等于分组中 ticket 的数值，以后当别的分组再次预约资源时就不需要真正的预约，只需要将该分组的 ticket 的数值加到资源的数量上即可，这样就保证了在一个组成员加入过程中资源只预约一份。当分组释放资源时，需要将资源的数量扣除该分组的 ticket 的数值，如果结果是 0，则释放资源。由于资源释放分组所经过的路径都是其预约资源时经过的路径，这样就保证了资源不会被不合时宜地释放。

3.2 信令格式

本算法在路由计算过程中传递的信令即控制分组格式如图 1。其中 ticket 是该分组需要被复制的数目，等于 1 时无须复制；class 是即将发送的数据流的服务等级；flag 用来标识该分组的类型，其值为 0 时是探测分组，为 1 时是回传

图 1 控制分组格式

分组，为 2 时是资源释放分组，为 3 时是确认分组；leavetree 用来标识该分组从源节点出发后是否到过非组播树中的节点，其值为 false 时表示没到过，为 true 时则到过；band 是业务请求所要求的带宽；hop 值表示分组从源节点到当前节点所经过的路由器的数量，但将该分组回传的路由器不算在内；max_hop 表示路径中所包含节点的最大数量，一旦有 hop=max_hop 且分组还未到达目的节点，则必须回传分组；cost 是分组所经过的路径的代价值；path 用来记录分组所经过的路径，以便在回传分组和发送确认时使用，并可用来防止分组经过不合理的路径到达目的节点。

探测分组(flag=0)：用于寻找路径并沿该路径进行接纳控制与软资源预约。

回传分组(flag=1)：回到上一级节点并重新进行路由。

资源释放分组(flag=2)：用于在释放分组所经过的路径上软预约的资源和所设置的相应信息。

确认分组(flag=3)：修改路由表，正式建立数据传输路径，并释放路径上软预约的资源和所设置的相应信息。

由于资源释放分组和确认分组的丢失会使算法付出的代价过高，故算法假定这两种分组均以可靠方式传送，即不考虑它们的丢失。

3.3 DQMTR 算法描述

(1)源节点 s 根据业务请求中的延迟、丢失率上限计算最大跳数 n，根据带宽需求计算所需的缓冲区数量，对各个出链路进行接纳控制，如果所有出链路的接纳控制都不成功则算法结束，否则软预约所有接纳控制成功的链路的资源，构造与接纳控制成功的链路的数量相同的控制分组，其中 flag 字段的值均为 0，即为探测分组，为所有分组设置相同的 ticket 值，并将相应链路中软预约的资源数量值标记为 ticket(实际上只预约了一份)，将源节点的 ID 加入路径信息，hop 数设为 1，分组中的 cost 加上相应链路的代价值，然后将这些分组发送出去。s 开始计时，如果收到成功消息，则算法结束，在时限内没有收到成功消息，则路由失败，结束。

(2)中间节点 i 对到来分组 P 的生存时间进行判断，如果小于等于 1，则置 flag=2，赋予其较大的生存时间，将其原路送回。否则，如果 P 为探测分组，按如下步骤进行。

步骤 1 判断 P 是否出现不合理路径，所经过的路径的跳数是否满足跳数约束。如果上述条件中任何一个为“真”，则置 flag=1，返回到上一级节点。否则，进行下一步；

步骤 2 检查 i 的除 P 的输入端口外的所有端口，将可用资源满足要求的端口加入候选端口集合 H，连接 i 和 i 在组播树中的下一级节点的链路及资源已经被别的探测分组预约过的链路在 i 中相应的端口可直接加入 H；如果 i 是组播树中的节点，设节点 j 也是组播树中的节点，但 j 不是 i 在组播树中的下一级节点，则连接 i 和 j 的链路所对应端口不能加入 H；

步骤 3 如果 H 为空，则置 flag=1，将 i 加入到 P 的路径信息中，将 P 返回到上一跳节点。否则，如果 P 的 tikcket 值 t 为 1，转步骤 4；若 t 大于 1，转步骤 5；

步骤 4 查找 i 的路由表得到 P 的下一跳端口 h。如果 $h\in H$，则记录 P 的下一跳端口为 h，否则取 H 中的任意一个作为 P 的下一跳端口；转步骤 6；

步骤 5 将 t 分成$|H|$个整数，其中$|H|$为 H 中元素的个数，使得这些整数的和仍为 t。设这些整数中非零整数的个数为 u，显然 $u\leqslant|H|$。构造 u 个探测分组，其 ticket 值分别与那 u 个非零整数相等。它们的下一跳端口取 H 中的 u 个元素。i 对每个新的探测分组都执行步骤 6；

步骤 6 设探测分组的下一跳端口为 h。如果对应的链路不在组播树中且资源没有被别的探测分组预约过，软预约相应的资源，将预约数量记录为分组的 ticket 值，将链路的代价加到分组的代价值中；如果资源已经被别的探测分组预约过，则将预约数量加上分组的 ticket 值，将链路的代价加到分组的代价值中。进行下一步；

步骤 7 如果下一跳节点不在组播树中，则标记分组的 leave 值为 1。将 i 加入到分组的路径信息中，发送分组。

如果 P 为回传分组，将 P 的输入链路中探测分组预约资源的数量减去 P 的 ticket 值，如果结果等于 0 则释放软预约的资源，否则将该结果更新预约资源的数量。P 的 cost 值也减去该链路的 cost 值，将 P 的 flag 值置为 0，再用与探测分组中 ticket 值等于 1 的分组相同的方法转发。

如果 P 为资源释放分组，将 P 的输入链路中探测分组预约资源的数量减去 P 的 ticket 值，如果结果等于 0 则释放软预约的资源，否则将该结果更新预约资源的数量。如果 i 是组播树中的节点，则将 P 丢弃；否则将 P 返回到上一跳节点。

如果 P 为确认分组，将 P 的上一跳节点标记为 i 在组播树中的下一级节点，修改路由表，将输入链路上软预约的资源更改为硬预约；如果 i 不是组播树中的节点，将 i 加入组播树；否则以可靠方式发送成功消息到源节点 s。

(3)目的节点 d 从接收到第 1 个探测分组时开始计时，等待 timeout 秒，保留在这段时间内到达的所有探测分组。等待结束后，分析在这段时间内到达的探测分组，从这些分组包含的路径中选择代价最小的路径，如果存在 2 个以上代价相同的路径则从中选出一个跳数最少的，将相应分组的 flag 值改为 3，其余分组的 flag 值改为 2，沿分组中保存的路径的反方向发送回去。在等待时间外到达的分组一律将其 flag 值改为 2，令其原路返回。

3.4 DQMTR 算法时间复杂度

算法的时间复杂度主要受以下因素的影响：ticket 的数量 t，分组丢弃概率 p，分组的最大跳数 h，网络中组播组数量 g，组成员数量 d。所以算法的最大循环次数可看作是这些元素的函数。本算法在一次路由计算过程中引入的开销由控制分组引入，控制分组引入的开销主要是组播树信息的查询、软预约资源信息的设置与查询导致的，分组构造与复制

的开销忽略不计。

一个探测分组或回传分组 P_1 在一个节点被丢弃的概率为 p 时 P_1 经历的跳数的期望为

$$\lim_{\text{ttl}\to\infty}\sum_{i=1}^{\text{ttl}} i\times(1-p)^i=(1-p)/p^2 \tag{8}$$

其中 ttl 是分组的生存时间。资源释放分组或确认分组 P_2 最多传送 h 跳，故 P_2 的最大跳数是 h。节点对每个探测分组都要检查其经过的路径以防止不合理路径的出现，由此引入的复杂度为 $O(h)$；查询组播树信息的时间复杂度为 $O(g)$；查询与修改软预约资源信息的时间复杂度为 $O(g)$。故 P_1 引入的时间复杂度为 $O((1-p)/p^2\times(h+g))$。P_2 引入的时间复杂度是 $O(h\times g)$。故每个控制分组在一次路由过程中引入的最大时间复杂度为 $O((1-p)/p^2\times(h+g)+h\times g)$。$P_1$ 和 P_2 的最大数量都是 t，故一次路由过程中引入的最大时间复杂度为 $O(t\times d\times((1-p)/p^2\times(h+g)+h\times g))$。

4　仿真实验设计及结果分析

实验所用的网络用 Waxman 方法生成的具有 100 个节点的随机网络，链路的带宽设置为 500–1000 之间的随机数。T 的源节点 s 和申请加入组成员节点都是随机选取的，其中成员节点数量取 10。算法中 ticket 的数值取 20。实验对比的性能参数有两个：成功率(success ratio)和平均路径代价(avg. path cost)，分别定义如下：

$$\text{success ratio}=\frac{\text{NCA}}{\text{TNCR}} \tag{9}$$

$$\text{avg. path cost}=\frac{\text{TCAECP}}{\text{NECP}} \tag{10}$$

其中 NCA 是成功加入的组成员数，TNCR 请求加入的组成员数；TCAECP 是成功建立连接的所有路径的代价之和，NECP 是成功建立连接的路径数量。每个实验值都是运行 100 次得到的平均值。实验中对比的算法是结合资源预留的分布式组播路由算法(记作 DQMTR)、传统的分布式组播路由算法(记作 TRADITATIONAL)和基于 TBP 算法[10]的组播路由算法(记作 MTBP)。其中，传统的分布式组播路由算法与本文中的结合资源预留的分布式组播路由算法的区别仅在于：分组在传送过程中不预约资源；TBP 算法也是一种基于 ticket 的分布式路由算法，其基本思想是将探测分组分成两种：黄色和绿色。黄色分组优化的是路径建立的成功率，其转发原则是哪个链路的可用带宽大就向哪个链路对应的端口转发；绿色分组优化的是路径的代价值，其转发原则是哪个链路的代价小就向哪个链路对应的端口转发。目的节点在多个可行路径中选择 1 个代价最小的建立连接。选择 TBP 算法是因为 TBP 是一种分布式多路算法，且能应用于非确定环境，这两点与 DQMTR 是相同的。只是 TBP 是一种单播算法，必须按照 DQMTR 的组播方法将其改造成组播算法 MTBP 后，才能与 DQMTR 算法进行比较。

图 2 是 3 种算法不同带宽需求条件下成功率的比较，这个实验中 3 种算法的跳数限制是 10。从中可以看出，带宽需求的增大，3 种算法的成功率都呈现下降趋势。而 DQMTR 的成功率则远高于 TRADITATIONAL 和 MTBP。这是因为 DQMTR 在路径探测过程中使用了资源预约，避免了确认分组在路径建立过程中因资源数量的变化导致建立失败的情况。

图 3 是这 3 种算法不同跳数约束条件下成功率比较，这个实验中 3 种算法的带宽需求是 200。从中可以看出，随着跳数限制的放宽，3 种算法的成功率都逐渐增大。但当跳数约束大于 4 后，3 种算法的成功率趋于稳定，这说明此时跳数限制对算法的成功率的影响很小。在跳数约束增大整个过程中，DQMTR 成功率都在 TRADITATIONAL 和 MTBP 之上。跳数约束较严格时，3 种算法的成功率都受跳数和带宽两种因素的影响，DQMTR 的优势不是很明显。当跳数约束放宽、算法摆脱跳数的影响后，可看出 DQMTR 的成功率大大高于 TRADITATIONAL 和 MTBP。而由于 MTBP 中黄色分组的作用，使得 MTBP 的成功率略高于 TRADITATIONAL。

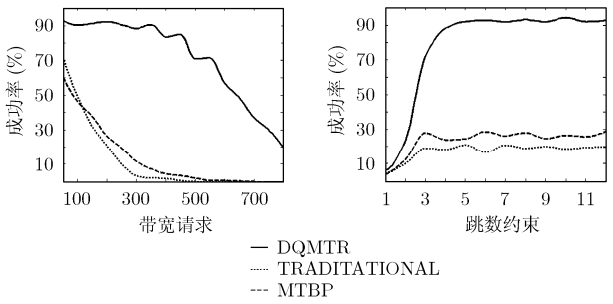

图 2 3 种算法不同带宽需求条件下成功率比较

图 3 3 种算法不同跳数约束条件下成功率比较

图 4 是算法 DQMTR、MTBP 不同带宽需求条件下平均路径代价的比较，这个实验中 3 种算法的跳数限制是 10。曲线 DQMTR 一直很平稳;带宽需求小于 700 时曲线 MTBP 略呈下降趋势，这是由于 MTBP 算法中绿色探测分组的作用。当带宽需求大于 700 后 MTBP 的代价值为 100，这意味着算法的成功率为 0。总的来看，DQMTR 在平均路径代价上的性能与 MTBP 相差不大，且在 MTBP 曲线优于 DQMTR 的区间(550–700)，MTBP 算法的成功率接近 0(见图 2)。从这个角度看，DQMTR 算法仍然优于 MTBP。

图 5 是算法 DQMTR，MTBP 不同跳数约束条件下平均路径代价的比较，这个实验中 3 种算法的带宽需求是 200。由图可知，在跳数约束小于 4 时，两种算法的代价都大体呈现上升趋势，此时 DQMTR 在平均路径代价上的性能与 MTBP 相比并无优势。当跳数约束大于 4 后，DQMTR 的优势比较明显，且两种算法的代价值大体稳定，因为此时算法已基本摆脱跳数约束的影响。图 5 说明在跳数限制较为宽松时，DQMTR 建立的路径比 MTBP 好。

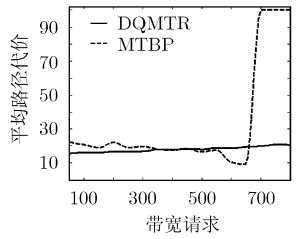

图 4 DQMTR 和 MTBP
不同带宽需求条件下
平均路径代价比较

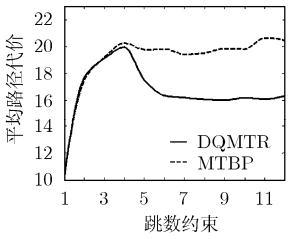

图 5 DQMTR 和 MTBP
不同跳数约束条件下
平均路径代价比较

5 结束语

针对 QoS 路由算法受网络环境的非确定性影响而导致性能下降的问题，本文提出了一种适用于 DiffServ 体系的结合资源预留的分布式组播路由算法。该算法通过在路径探测阶段预约资源，避免了在路径建立阶段因资源数量的变化导致建立失败的问题，并使用资源纪录解决了资源的过预约问题。实验表明，该算法在组成员加入成功率和平均路径代价上表现出了一定的优越性。

参 考 文 献

[1] Lorenz D H and Orda A. QoS routing in networks with uncertain parameters. *IEEE/ACM Trans. on Networking*, 1998, 6(6): 768–788.

[2] 朱慧玲, 杭大明. QoS 路由选择: 问题与解决方法综述. 电子学报, 2003, 1(31): 109–116.

[3] Lorenz D H and Orda A. QoS routing in networks with inaccurate information: Therory and algorithms. *IEEE/ACM Trans. on Networking*, 1999, 7(3): 350–364.

[4] Yan Xin and Li Layan. Distributed QoS multicast routing in networks with imprecise state information. *Joural of Systems Engineering and Electronics*, 2005, 16(4): 866–874.

[5] 陈萍, 董天临, 石坚. 一种基于概率的 QoS 单播路由算法. 软件学报, 2003, 14(3): 582–587.

[6] Lorenz D H and Orda A. Optimal partition of QoS requirements on unicast paths and multicast trees. *IEEE/ACM Trans. on Networking*, 2002, 10(1): 102–113.

[7] 林闯, 单志广等. 计算机网络的服务质量(QoS). 北京: 清华大学出版社, 2004: 35–47.

[8] 李谢华. 分布式 QoS 路由算法的研究. [硕士论文], 中南大学, 2004.

[9] Song Jun and Pung Hung Keng. Fast and efficient flooding based QoS routing algorithm. IEEE Conference on Computer Communication and Networks, Boston, USA, 1999: 298–303.

[10] Chen Shigang and Nahrstedt K. Distributed QoS routing with imprecise state information. IEEE International Conference on Computer Communicatons and Networks, L.A., USA, 1998: 614–621.

杜 荔： 女，1962 年生，副教授，博士，研究方向为宽带通信领域中的 IP 交换技术.

刘玉涛： 男，1984 年生，硕士生，研究方向为 QoS 组播路由.