一种基于 OSPF 扩展的预计算 QoS 路由算法研究*)

张 静 冉晓旻 胡捍英
(解放军信息工程大学 郑州 450002)

摘 要 在一个 MPLS 域,LSPs 的建立需要 QoS 路由协议分发 QoS 相关的信息和执行 QoS 路径选择,但是传统的 OSPF 不支持 QoS 路由。本文提出并详细讨论了一种 OSPF-QoSR 路由机制,它是对 OSPF 路由协议的扩展,基于网络的动态可用带宽资源和流的 QoS 请求来决定流的 QoS LSPs。仿真证明,该机制在丢包率、链路利用率、延时方面的性能优于只考虑最短路径的 OSPF。

关键词 MPLS,QoS 路由,OSPF,OSPF-QoSR

Research on an OSPF-based Pre-Computational QoS Routing Extension Algorithm

ZHANG Jing RAN Xiao-Min HU Han-Ying
(Information Engineering University, Zhengzhou 450002)

Abstract Within a MPLS domain, the setting up of LSPs needs a QoS-aware routing protocol for the dissemination of QoS-related information and the actual QoS path selection. While the original OSPF does not support QoS routing, the OSPF-QoSR routing scheme which is extensions of OSPF has been proposed and discussed in detail in this paper. QoS LSPs for flows are determined based on some knowledge of dynamic bandwidth resource availability in the network as well as the QoS requirements of flows. Simulation results show that the OSPF-QoSR scheme outperforms the OSPF scheme considering only the shortest paths in terms of loss ratio, link utilisation and delay.

Keywords MPLS, QoS routing, OSPF, OSPF-QoSR

1 OSPF 路由现状

普通路由协议通常只用源和目的间的跳数度量来计算最短路径,而不能根据应用的带宽、延时、延时变化等要求来提供路由。比如,OSPF 是链路状态路由选择算法,面对大型网络时,能对网络拓扑、路由信息变化做出快速反应并较快收敛,具有支持负载平衡、变长掩码等特性,所以 OSPF 应用非常广泛。但是由于 OSPF 等路由协议的拓扑驱动和最短路径路由的本质,网络在高负载情况下可能出现一些链路高度拥塞(R3-R4-R5),而其它链路的利用率较低(R7-R8)的情况,即著名的"鱼形问题"(如图 1),从而导致较差的 IP QoS。因此,开发支持目前和未来各种 Internet 服务的 QoS 路由协议,有效地管理和利用网络资源,成为网络发展的一个重要方向。

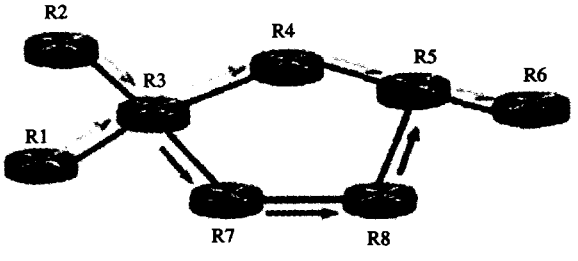

图 1 鱼形问题

理论上,链路状态协议可支持多种不同的路由度量,路由选择可基于不同的 QoS 属性,而不需要改变拓扑计算的时间。旧版 OSPF[6] 规范就支持基于 IP 包的 5 个不同服务类型的 TOS 路由,路由器可以通过 SLAs 通告 TOS 开销度量,并计算多种 TOS 的特殊路由表。应用通过设置 IP 包头的 TOS 域来提出 QoS 要求,路由器根据 TOS 把该应用的数据包路由到目的。但是,实际上由于缺乏应用,新版 OSPF[7] 规范已经去除了关于 TOS 路由的描述。

OSPF 应用广泛,所以在 OSPF 的基础上增强其 QoSR (QoS Routing)能力,易于实现和推广。本文详细分析和研究的一种 QoS 路由算法:OSPF-QoSR,就是对 OSPF 协议的 QoS 能力的扩展。通过在 MPLS 网络环境下仿真,得出结论:OSPF-QoSR 可以改善网络的包丢失率、链路利用率和包延时等性能。

2 OSPF 的 QoS 扩展原则、目标及思路

设计 OSPF-QoSR 协议的基本原则是既要使 OSPF 支持 QoS,又要对现存的 OSPF 算法的改动最小、减小额外的开销且易于实现;OSPF 的 QoSR 扩展的目标是为 QoS 流提高性能,而且要向后兼容。对当前 OSPF 协议及其实现的影响最小。由于 QoS 路由本身的复杂性,要达到此目的,显然意味着要在"最优性"和"简单性"之间权衡。扩展思路有两点:

- 增强链路状态通告和拓扑数据库,使其包含网络资源信息(如可用带宽)。
- 考虑网络资源信息,采用可选的路由计算算法来计算路由。

3 OSPF 协议的扩展

3.1 OSPF options 域扩展

OSPF 的 Hello Packets, Database Description packets 和所有 LSAs 都包含 options 域字节[6],如图 2。

*)总装基金项目:"智能计算机网络安全技术研究"(编号:151415020101JB5201)。张 静 副教授;冉晓旻 副教授,研究方向:通信与信息系统;胡捍英 教授、博士生导师、863 移动通信组专家组成员,研究方向:无线与移动通信技术专业。

*	*	DC	EA	N/P	MC	E	T

图 2 OSPF options 八位组

T-bit 表示 TOS 路由能力，为向后兼容。扩展后的 T-bit 用来指示路由器的 QoS 能力，并重新命名为 Q-bit。在 Hello packets 中，该比特指示路由器是否能支持 QoS 路由；在 Router LSA 或 Summary links LSA 中设置该比特，说明数据包含 QoS 域；在 Network LSA 中，该比特指出通告中描述的网络是否能支持 QoS。

3.2 OSPF TOS 域扩展

在语义上，OSPF-QoSR 的“QoS 编码”兼容并扩展了 OSPF 的“TOS 编码”。除了保留 4 位长的 OSPF TOS 定义，QoS 资源编码启用了包含在不同 LSAs 的 TOS 域的第 5 位，即有 32 种组合。此外，考虑到 OSPF 路由器并不能理解 OSPF-QoSR TOS 域的所有 bits，所以 OSPF-QoSR 目前只定义了带宽和延时度量。这样，即使 OSPF 路由器不考虑最高有效位，带宽和延时仍可以分别被映射为“最大吞吐量”和“最小延时”，达到了对现有 OSPF 协议及其实现影响最小的目的。

3.3 OSPF metric 域扩展

OSPF 包的 metric 域只提供 16bits 来编码度量值。如果采用可用资源的线性表示，链路支持带宽到 GBytes/s 是不可行的。解决方法是采用指数编码：选择合适的隐含基数、尾数和指数的数位。

• 带宽编码规则：用 16 位表示可用带宽，其中 3 个最高有效位是指数(假设基数是 8)，剩下的 13 位做尾数，那么可用带宽的值为：(2^13−1) * 8^x Bytes/s(x 是指数值)。通告上述可用带宽二进制编码的补码，即通告值是：2^16−1-(指数编码的可用带宽)。这样设计的目的是把带宽度量和链路的开销(cost)度量一致起来，从而可兼容使用 OSPF 路由器的 Dijkstra 算法来计算最低开销路径。

• 延时编码规则：延时编码使用类似带宽编码的指数方法，假设基数为 4，那么延时值为：(2^13−1) * 4^xμs(x 是指数值)。

4 OSPF-QoSR 链路状态信息通告

每个路由器都维护一个网络拓扑数据库，库中包含每条链路的当前状态(可用带宽和传播时延)。链路度量的更新需要在网络中通告，以便每个路由器计算准确一致的 QoS 路由。OSPF-QoSR 的链路状态信息通告是基于扩展的 OSPF 协议格式本文第 3 节。频繁更新网络状态信息的开销和周期更新网络状态信息的准确度之间需要权衡，因此 OSPF-QoSR 提出 4 种链路状态更新策略。

4.1 周期更新(period based updating, PB)

PB 算法的基本思想就是通过设置一个固定的超时定时器，周期更新网络拓扑。该算法虽然可直接控制通信开销，但不能保证及时传播状态改变，尤其是当超时定时器设置较大值时。

4.2 阈值更新(threshold based updating, TB)

TB 算法的基本思想是设置一个固定的阈值 T，对于节点的接口 i，bw_i^p 表示其最新通告的可用带宽值，bw_i 是其当前可用带宽值。当($|bw_i^p - bw_i| / bw_i^p$)$>T$($bw_i^p>0$) 时就触发更新；特别是当 $bw_i^p=0$ 时，总是触发更新。因此，当链路工作在低可用带宽范围时，TB 算法可以通过较频繁更新提供更及时准确的状态信息，而可用带宽值越大，状态信息的准确度越低。

4.3 等宽类更新(equal class based updating, ECB)

ECB 算法的基本思想是通过设置约束值 B，把一条链路的可用带宽划分成多个等宽的类：$(0,B)$，$(B,2B)$，$(2B,3B)$ …。当一个接口可用带宽的变化跨越类边界时，就触发一次更新，使更新后的链路带宽值属于一个与更新前不同的类。由于类是等宽的，所以 ECB 算法对链路带宽的所有分类都有相同的状态描述准确度。

4.4 非等宽类更新(unequal class based updating, UCB)

UCB 算法的基本思想是设置两个常量 B 和 $f(f>1)$，用于定义大小不同(非等宽)的类：$(0,B)$，$(B,(f+1)B)$，$((f+1)B,(f^2+f+1)B)$，$((f^2+f+1)B,(f^3+f^2+f+1)B)$，…。类的宽度根据因子 f 呈几何增长。类似 ECB，当带宽变化超越类边界时，就触发更新。该策略在高可用带宽范围中有较少的宽度较大的类，而在低可用带宽范围中有较多的宽度较小的类。也就是说，UCB 对于低可用带宽范围有更细致和准确的状态描述。

从上看出，只有当链路的当前可用带宽与上次通告的值有明显差别时，TB、ECB 和 UCB 才触发更新。此外，TB、ECB 和 UCB 算法的更新范围扩展到节点的所有链路，即使是由一条链路触发的更新，节点所有接口的可用带宽值都要通告，这与 OSPF 路由协议在节点所有关联链路上产生 LSUs 的行为相同。

无论是基于阈值，还是基于分类的 LSU 方法，都在状态信息准确度和更新量之间取得权衡。但是，当出现快速流量抖动时，可能会触发频繁更新，结果造成瞬间的控制开销。为缓解该问题，OSPF-QoSR 引入抑制定时器(hold-timer)来强制 TB、ECB 和 UCB 连续更新之间的最小时间间隔。

5 OSPF-QoSR 路由算法

5.1 网络模型和求解目标

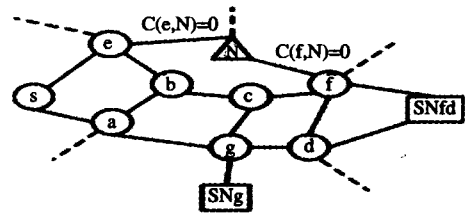

图 3 网络模型拓扑

考虑任意给定的网络 $G(V,E)$，如图 3，其中 $V(G)$ 是网络中所有顶点的集合，顶点数为 N；$E(G)$ 是任意两相邻顶点 a 和 b 间的通信链路(a,b)的集合；$B(a,b)$表示链路$(a,b)\in E(G)$的剩余可用带宽；$C(a,b)$表示链路$(a,b)\in E(G)$的开销；$ID(v)$表示路由器顶点 $v\in V(G)$ 的标识符(可用 IP 地址标识)。$BW(P(s,d))$表示路径 $P(s,d)$的瓶颈带宽；W_{min}表示应用所需的最小带宽。

根据以上建立的网络模型和 QoS 要求，我们将 OSPF-QoSR 算法的求解问题描述为：给定网络 $G(V,E)$、信源 s、信宿 d、寻找信源 s 到信宿 d 的路径 $P(s,d)$，路径 $P(s,d)$必须满足以下条件：

1) $BW(P(s,d))\geqslant W_{min}$

2) $\sum_{(a,b)\in P(s,d)} C(a,b)\rightarrow \min$

5.2 算法前提与假设

OSPF-QoSR 路由器使用扩展的 BF(Bellman-Ford)算法，就是要找一个能支持一个请求的连接的、跳数最少的低开

销路径。在讨论算法之前，假设：

(1)在路由选择之前，节点已经知道到目的 d 的流的 QoS 请求：$W_{min}=B$。

OSPF-QoSR 在路径选择时只考虑带宽请求，因为 A. Orda and A. Sprintson[8] 证明了当一个连接有多个 QoS 要求时，可以把它们转换为带宽要求。Guerin 等人[9] 也曾提出一个等式把延时约束映射为带宽约束。此外，带宽度量在大多数情况很合适，能保持一个可接受的计算复杂度。其它 QoS 参数，如延时和延时抖动被忽略。在执行路径计算之前，延时参数一般用来剪枝高延时链路(如卫星链路)。

(2)对于 $\forall v\in V$，节点 v 能通过扩展 OSPF 协议中的带宽和延时编码的 LSAs，获得网络最新全局状态信息，包括：

• 网络的最新拓扑信息；

• 对于 $\forall (a,b)\in E$，链路 (a,b) 的可用带宽状态信息：$B(a,b)$。

(3)算法采用预先计算形式，计算到所有目的的、所有可能请求带宽的路径，这是为了弥补按需计算(on-demand computation)中按请求触发路径计算开销大的不足，从而保持 QoS 路由计算开销在一个与传统路由算法可比的等级。但预先计算模式比按需计算模式的每次计算量要大。

在以上假设的前提条件下，OSPF-QoSR 路由算法需要完成的任务就是：预先计算 QoS 路由表，表中包含到所有目的的、所有跳数的最大剩余可用带宽的路径信息；采用 WSP (Widest-Shortest Path)路径选择机制，从与 OSPF 路由表分离的 QoS 路由表中，提取已知目的和带宽请求的流的转发信息。

5.3 路由表结构设计

源节点 S 的 OSPF-QoSR 路由表是一个 $N\times H$ 矩阵，N 是目的顶点数，H 是路径的最大可允许(或可能的)的跳数。H 的取值可以是网络的直径，或显式设置为限制路径最多只能 H 跳，并且此时的 H 值必须大于图 3 中到任意节点的最少跳数的路径长度。路由表项 $\langle d_i,h\rangle$ $(i=0\cdots N-1, d_0=s; h=0\cdots H-1)$ 包含两个域：

• bw(bandwidth)：表示在源节点 s 到目的节点 d_i 之间最多 h 跳的路径上的最大可用带宽。

• ri(routing information)：该域显示的是剩余可用带宽为 bw、最多 h 跳到目的节点 d_i 的路径的相关的路由信息。在逐跳路径选择中，ri 域是该路径上与源节点 s 邻接的下一跳节点。该节点一般是一个路由器，唯一例外的情况是，网络就是目的节点，被选择的路径恰好是一条边，把源节点连入该网络。在显式路径选择中，ri 域是该路径上前一跳节点的 ID。

5.4 算法思想

假定源节点 s 调用 OSPF-QoSR 路由算法来计算本地路由表。

首先初始化路由表，表中所有的 bw 域设为零，并清空 ri 域。

接下来，路由器 s 的邻接点的第一列(即一跳路径)修改如下：bw 域设为由源 s 发出的、与源 s 直接相连的边上的可用带宽值；ri 设为路由器 s 的邻接路由器 ID。

然后，算法要经过最多 H 次循环来计算到所有可能目的节点的、所有可能跳数的路径。在第 h 次循环中，首先要把第 $h-1$ 列拷贝到第 h 列，同时算法保持一个列表，其中包含在上次循环中(即 $h-1$ 次循环)bw 域值改变的节点：

$v_1, v_2, v_3, \cdots, v_k (k\leqslant N)$

对列表中的每个节点 v_j $(j=1\cdots k)$，算法分别查看链路 (v_j,m) 所连接的 v_j 的所有邻接节点 m，检查到节点 m 的最多 h 跳路径的最大可用带宽。在表项 $\langle v_j,h-1\rangle$ 的 bw 域和拓扑数据库中保持的链路度量值 $B(v_j,m)$ 中取最小值，如果 $\min(bw\langle v_j,h-1\rangle, B(v_j,m))>bw(m,h)$，那么就会找到到目的节点 m 的 h 跳的一条更好(即剩余带宽值更大)的路径，于是更新表项 $bw\langle m,h\rangle=min(bw\langle v_j,h-1\rangle, B(v_j,m))$。只要更新了表项 $\langle m,h\rangle$，该项 ri 的域也要根据不同的路由策略进行相应的修改：

• 在逐跳路由中，修改表项 $ri\langle m,h\rangle=ri\langle v_j,h-1\rangle$。$ri\langle m,h\rangle$ 记录了 h 跳到目的节点 m 的最佳路径上的第一跳(即从源开始的下一跳)。

• 在显式路由中，修改表项 $ri\langle m,h\rangle=ID(v_j)$。$ri\langle m,h\rangle$ 记录了 h 跳到目的节点 m 的最佳路径上的前一跳。

源节点 s 调用 OSPF-QoSR 算法经过上述的 H 次循环，计算出到所有目的节点的最多 H 跳的 QoS 路由表。

5.5 特殊问题及解决方法

(1)“cost=0 链路”问题

现实中存在路由器通过传输网络(如：ATM、Ethernet 等)互连的情况，比如考虑 2 个路由器 e,f，通过以太网 N(如图 3 中的三角形顶点)连接。网络 N 连接的两个路由器 e,f 间属于同一个物理跳，应看作是一个单跳路径，即网络 N 的入边和出边中只有一个在跳数计算中记数。所以 OSPF-QoSR 算法有必要考虑“cost=0 链路”的处理。

• 前提：网络 N 到 e,f 的 cost 都记为 0；

• 在每次循环 h 中$(1\leqslant h<H)$，一旦项 $\langle v_j,h\rangle$ 被修改，就检查是否有 cost=0 的链路 (v_j,m) 从点 v_j 发出(此时 v_j 是一个传输网络)。

• 在当前循环中，修改节点 m 的项 $\langle m,h\rangle$(而不是修改项 $\langle m,h+1\rangle$，因为边 (v_j,m) 并没有增加跳数)。如果 $\min(bw\langle v_j,h\rangle, B(v_j,m))>bw\langle m,h\rangle$，那么就修改项 $\langle m,h\rangle$ 的 bw 域为：

$$bw\langle m,h\rangle=\min(bw\langle v_j,h\rangle, B(v_j,m))$$

• 在逐跳路由中，$ri\langle m,h\rangle=ri\langle v_j,h\rangle$；在显式路由中，$ri\langle m,h\rangle=ID(v_j)$。

(2)残桩网络问题

残桩网络(stub network)是只有一个出口连接到路由器的网络，通常报文的源地址和目的址都在残桩网络中。如图 3 中的矩形顶点，SN_{fd} 是出口分别连接到路由器 f 和 d 的残桩网络。为了降低计算复杂度，可采用后处理方式解决残桩网络问题。

• 前提：由每个连接残桩网络的路由器 v 通告到残桩网络 SN_v 的可用带宽 $B(v,SN_v)$。

• 计算到残桩网络的可用 QoS 路由，就是扩展已经建立的 QoS 路由表。首先检索出连接残桩网络的路由器，然后在 QoS 路由表中为每个残桩网络顶点增加一行，新行的列对应不同跳数的路径，同样包含带宽和路由信息。对应残桩网络行的每一项初始化为 0，ri 域设为 null。对应于残桩网络 SN_v 的新行第 h 列的 $bw\langle s_v,h\rangle$ 值修改如下：

$$bw\langle SN_v,h\rangle=\max(bw\langle SN_v,h\rangle, min(bw\langle v,h\rangle, B(v,SN_v)))$$

也就是说，只有当 h 跳路径到路由器 v 的带宽和路由器 v 与 SN_v 间链路带宽的最小值大于当前 h 跳到残桩网络 SN_v 的带宽值时，才修改 $bw\langle SN_v,h\rangle$。

• 当经过路由器 v 的带宽大于或等于当前值时，修改 SN_v 的 ri 域：大于时，用相应列路由器 v 的 ri 域代替 SN_v 的 ri 域的当前值；等于时，把相应列路由器 v 的 ri 域集合加入

到 SN_v 的 ri 域集合。

(3)相同开销问题

OSPF-QoSR 算法可能会遇到相同开销问题:存在到同一目的 d 的,等跳数、等带宽的多条路径。要解决该问题,可以把表项(d,h)的 ri 域扩展为一个列表,同时记录多条路径的下一跳 ID。

如果 OSPF-QoSR 算法存储了多条相同开销的路径,可以考虑以下几种方法进行路径选择:

- 用轮询方式在等开销路径列表中选择下一跳;
- 根据本地接口的实际可用带宽加权的概率值来选择下一跳,以均衡负载。

5.6 算法伪码描述

```
OSPF-QoSR(G,V,E,s,H)
{
 for (每个顶点 d∈V)
 {
  RT[d,0]. bw = 0; RT[d,0]. ri=null;
  RT[d,1]. bw = 0; RT[d,1]. ri=null;
 }/*初始化*/
 RT[s,0]. bw =∞;
 reset prevlist;/* prevlist 是上次循环中域值改变的顶点列表*/
 for (s 的所有邻接顶点 n)
 {
 RT[n,1]. bw=B[s,n];
 RT[n,1]. ri =ID[n];
 prevlist = prevlist union {n};
 for (每个传输网络顶点 v∈prevlist)
  for (每条cost=0 的链路(v,m)∈E ) /* h=1 时,"cost=0 链路"
          问题的处理*/
  if (min( RT[v,1]. bw,B[v,m])>RT[m,1]. bw)
    {
    RT[m,1]. bw=min( RT[v,1]. bw,B[v,m]);
    RT[m,1]. ri=RT[v,1]. ri; /*在显式路由中,RT[m,1]. ri:=
                          ID(v)*/
    }
 }/*修改路由表 RT[1..N,1..H]的第一列*/
 for(h=2;h<H;h++) (*1)
 {
  reset newlist;/* newlist 是本次循环中域值改变的顶点列表*/
  for (每个顶点 d∈V)
  {
   RT[d,h]. bw=RT[d,h-1]. bw;
   RT[d,h]. ri=RT[d,h-1]. ri;
  }
  for (每个顶点 v∈prevlist)  (*2)
   for (每条链路(v,m)∈E)  (*3)
   {
   if (min( RT[v,h-1]. bw,B[v,m])> RT[m,h]. bw)
     {
     RT[m,h]. bw=min(RT[v,h-1]. bw,B[v,m]);
     RT[m,h]. ri=RT[v,h-1]. ri; /*在显式路由中,RT[m,h].
                          ri:=ID(v)*/
     newlist=newlist union {m};
   }
  if (v 是传输网络顶点 && C[v,m]==0)/* 2≤h<H 时,"cost
     =0 链路"问题的处理*/
   if(min( RT[v,h]. bw,B[v,m])>RT[m,h]. bw)
   {
     RT[m,h]. bw=min(RT[v,h]. bw,B[v,m]);
     RT[m,h]. ri=RT[v,h]. ri; /*在显式路由中,RT[m,1]. ri:
                          =ID(v)*/
     newlist=newlist union{m};
     }
   }
  prevlist=newlist;
  if(prevlist==null) h=H+1;/*如果上次循环中没有域值改变
     的节点,就跳出循环*/
 }
 for (每个顶点 d∈V)/*后处理到残桩网络的 QoS 路由*/
  if (d 连接残桩网络 SNd)
    for(h=0;h<H;h++)
    {RT[SNd,h]. bw =0; RT[SNd,h]. ri=null;}/*路由表 RT 中
                残桩网络顶点项的初始化*/
 for(每个顶点 d∈V)
   if(d 连接残桩网络 SNd)
   for(h=1;h<H;h++)/* 路由表 RT 中残桩网络顶点项的修改
                */
   if(min( RT[SNd,h]. bw,B[d,SNd])> RT[SNd,h]. bw)
 {
  RT[SNd,h]. bw=min(RT[SNd,h]. bw,B[d,SNd]);
  RT[SNd,h]. ri=RT[d,h]. ri;/*在显式路由中,RT[SNd,h]. ri=
                          ID(d)*/
 }
 return RT;/*算法返回的数组 RT 就是为源 S 预先计算出的 QoS
          路由表*/
}
```

5.7 算法复杂度分析

假设不考虑残桩网络情况,OSPF-QoSR 算法的计算复杂度主要由伪码中的(*1)的 for 循环所决定。该 for 循环的最大循环次数为 $H-2$,那么对于每一跳的循环来讲,上一跳循环中域值改变的节点数最多为 N(即(*2)的 for 循环次数)。而对于每个域值改变的节点,最多有 $N-1$(即(*3)的 for 循环次数)条链路要被查看,因此 OSPF-QoSR 算法最坏情况的计算复杂度为 $O(HN^2)$。

5.8 路由转发信息提取

当源 s 运行 OSPF-QoSR 算法预先计算完本地路由表,就可以为到达源 S 的、已知目的 d 和带宽要求为 B 的流从路由表中提取相应的转发信息。

- 逐跳路由的情况比较简单,只需返回可行路径的下一跳。因此,就以目的顶点 d 为索引,在 s 的 QoS 路由表中找到相应行,然后随着跳数的(列号)增加来检索,直到找到某项$\langle d,h_d\rangle$,满足 $bw\langle d,h_d\rangle \geqslant B$,那么 $ri\langle d,h_d\rangle$就是为该流返回的、最少 h_d 跳就能满足流的 QoS 请求的最佳路径的下一跳路由器 ID。

```
HR-Nexthop(RT,s,d,B)
{
for (h=0;h<H;h++)
  if (RT[d,h]. bw>=B) break;
  return RT[d,h]. ri;
}
```

- 在显式路由中,需要从 QoS 路由表中为流提取出可行的转发路径。其实这是一个循环过程,如上所述,先找到某项$\langle d,h_d\rangle$,满足 $bw\langle d,h_d\rangle \geqslant B$ 时,那么 $ri\langle d,h_d\rangle$就是目的 d 的前一跳。再以 $ri\langle d,h_d\rangle$为目的,找到其前一跳:$ri\langle ri\langle d,h_d\rangle, h_d-1\rangle$。如此重复执行该逆向检索过程,直到到达路径的源 S。

```
ER-Path(RT,s,d,B)
{
 for (h=0;h<H;h++)
 if (RT[d,h]. bw>=B)
 {
     PATH[h]=d;
     prehop=RT[d,h]. ri;
 while(prehop! = s)
   {
    h=h-1;
    PATH[h]=prehop;
    prehop=RT[prehop,h]. ri;
   }
   PATH[0]=s;
   break;
 }
return PATH
}
```

6 仿真结果及分析

本文采用离散事件网络仿真器 JaNetSim 建立仿真环境。对提出的 OSPF-QoSR 与 OSPF 机制进行了比较,包括普通开销度量的 OSPF(OSPF-flat)、以链路容量的倒数(1/cap)作为开销度量的 OSPF(OSPF-invcap)。OSPF-invcap 是 Cisco 的常用机制。所有算法在一个 MPLS 环境,使用信元交换的 ATM 标记交换路由器(ATM-LSRs),采用分布式的按需路由,通过 CR-LDP 标记分配协议,为 EF-class 通信预留路径带宽。

仿真拓扑结构是基于典型的 MCI Internet 骨干(如图 4),该拓扑包括 18 个路由器和 32 条链路,链路带宽有 3 个级别的容量,低、中、高容量链路的带宽分别是 4 Mbit/s,5 Mbit/s,7 Mbit/s。为减小仿真量,带宽根据实际值按比例缩小。

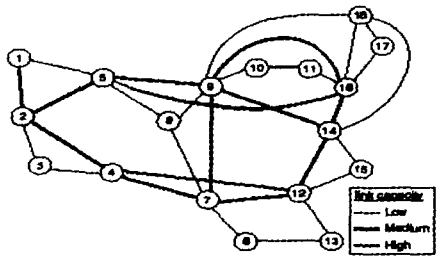

图 4　仿真拓扑

仿真设定每个路由器连接一个客户站点，站点支持 6 个通信类，包括 EF、4 个 AF 和 1 个缺省的 DiffServ PHB，每个通信类有 36 个源，所以共有 216 个通信源。信元到达时间间隔符合泊松分布。对上述 3 种路由机制都执行 12 个仿真会话，每个会话通过增加通信源的平均传输率来增加网络负载，从一个非拥塞的网络到一个轻微拥塞的网络，每个仿真会话运行 180s。为了充分测量每种路由机制的性能，每个通信源的寿命以实际值按比例缩小。这样，仿真期间，可以大量地建立或释放 LSP 事件。

6.1　丢包率性能仿真

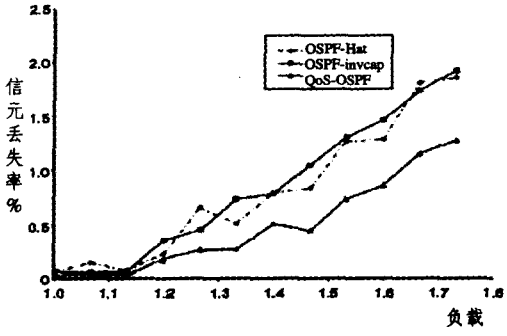

图 5　三种路由机制的信元丢失率

图 5 显示了每个路由机制的信元丢失率。OSPF-flat 和 OSPF-invcap 的信元丢失率最高，当网络负载增加时，OSPF-QoSR 比前二者有明显较低的信元丢失率。因为前两种机制都采用静态开销度量，不能适应链路剩余带宽的变化，因此考虑剩余带宽的动态路由机制 OSPF-QoSR 优于采用静态开销度量的机制。

6.2　链路利用率性能仿真

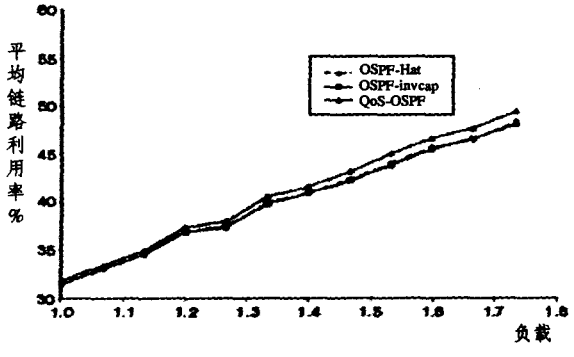

图 6　三种路由机制的平均链路利用率

流量工程的目的之一是优化网络资源利用率，仿真测量了 3 种路由机制的平均链路利用率。图 6 显示：OSPF-QoSR 机制有最高的平均链路利用率，OSPF-flat 的最低。可见，把剩余可用带宽作为动态路由参数的 OSPF-QoSR 机制，其提高网络资源利用率的性能要高于采用静态开销度量的路由机制。

6.3　延时性能仿真

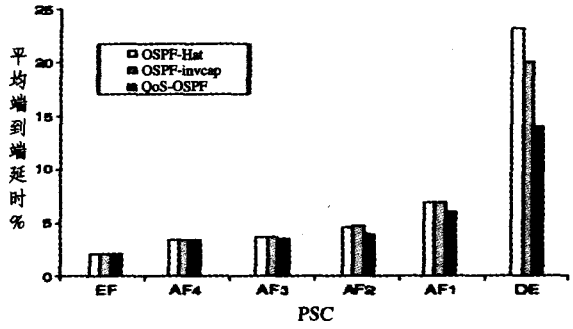

图 7　三种路由机制的平均端到端延时(低负载)

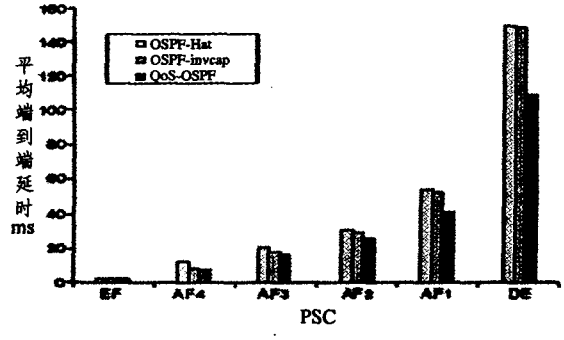

图 8　三种路由机制的平均端到端延时(高负载)

图 7 显示了在一个相对较低负载的网络下，3 种路由机制的每个 DiffServ PSC(PHB Sheduling Class，是指一类具有相同队列处理要求的 PHB)的信元的平均端到端延时。图 8 显示了在一个高负载网络(两倍于低负载)下的同样信息。可见，无论网络负载的高低，总体上，静态开销度量机制(OSPF-flat 和 OSPF-invcap)比 OSPF-QoSR 机制的端到端延时大。

结束语　随着 Internet 的发展和商业化，QoS 显得尤为重要。为支持 QoS 路由，本文提出的 OSPF-QoSR 路由机制，就是通过路由器接收和广播带有带宽和延时编码的 LSAs，获取动态网络状态及可用资源信息，再用考虑带宽约束的扩展 BF 算法，计算输出一个满足约束的 LSR 地址的最短路径的显式路由。该显式路由传递给信令协议，由信令部分完成从源到目的的 LSP，以及沿途 LSR 转发状态的建立。通过仿真比较，证明 OSPF-QoSR 路由机制在丢包率、链路利用率、延时方面的性能，优于传统的 OSPF 算法。

参 考 文 献

1　Faucheur F, Wu L, Davie B, et al. MPLS support of differentiated services. Work in progress(Internet draft), Aug. 2000
2　Faucheur F, Nadeau T D, Chiu A, et al. Requirements for support of Diff-Serv-aware MPLS traffic engineering. Work in progress (Internet draft), Nov. 2000
3　Faucheur F, Nadeau T D, Chiu A, et al. Extensions to RSVP-TE and CR-LDP for support of Diff-Serv-aware MPLS traffic engineering. Work in progress(Internet draft), Nov. 2000
4　Crawley E, Nair R. A Framework for QoS-based Routing in the Internet. RFC 2386, August 1998
5　Apostolopoulos G, Williams D. QoS Routing Mechanism and OSPF Extensions. RFC 2676, August 1999
6　Moy J. OSPF Version 2. RFC 1583, March 1994
7　Moy J. OSPF Version 2. STD 54, RFC 2328, April 1998
8　Orda A, Sprintson A. QoS Routing: The Precomputation Perspective. IEEE INFOCOM 2000-The Conference on Computer Communications, 2000, 1: 128～136
9　Guerin R, Orda A, Williams D. QoS Routing Mechanisms and OSPFextensions. IETF Internet Draft, 1996. citeseer. nj. nec. com / guerin96qos. html
10　Singh A, Mittal G. QoS and Traffic Engineering: MPLS, DiffServ and Constraint Based Routing. A project report done in partial fulfillment of the requirements for the degree of Bachelor of Technology, May, 2000