层次网络中的拓扑压缩算法及性能比较

Topology Aggregation Algorithm in Hierarchy Network and Performance Comparison

杨 敏 向 勇 史美林 陆慧梅

(清华大学计算机科学与技术系 北京 100084)

Abstract Future Internet will be a hierarchy topology. To guarantee peer-to-peer QoS, QoS-based routing protocol will be adopted. Considering scalability and security, topology information used by routing protocol should be compressed before distribution in network. Routing protocol would route the packets without knowing the complete network topology information. This article investigates several different topology aggregation schemes and their application area. We also compare the performance of different topology aggregation schemes under different topology structures and update policies.

Keywords Hierarchy network, Topology aggregation, Performance comparison

随着 Internet 规模的迅速扩大,QoS 路由面临复杂度过高的问题。在大规模网络中实现 QoS 路由的主要困难在于链路 QoS 信息的频繁更新和 QoS 最优路径的计算。链路 QoS 信息(带宽和时延)处于不断变化的状态,需要将这些变化及时地扩散出去以使路由程序做出正确的计算。然而频繁地更新 QoS 信息无疑会增加网络负载,降低了可扩展性,因此尽量减少链路 QoS 信息的更新对于提高 QoS 路由算法的可扩展性起到至关重要的作用。

减少链路 QoS 信息更新的措施包括:减少信息量和减少更新频率两个方面,其中减少信息量可通过减少需要扩散的信息以及减少信息的扩散范围来实现。拓扑压缩和网络层次结构划分是重要的减少信息量方法。假定边界路由器的数量为 n,拓扑压缩通过一定的算法,将需要扩散的 QoS 信息量减少到与 $O(n^2)$甚至 $O(n)$一个数量级。层次结构则通过分层将低层的 QoS 信息限制在本区域内扩散而不用向高层扩散,有效地降低了网络负担。

本文将概述层次结构中的各种拓扑压缩算法并对它们的性能进行比较。

1. 层次结构

正如字典和电话簿中的作法一样,在大规模的网络中采用层次结构对提高效率有很大帮助。在层次结构的网络中,网络节点被分成若干组[18],与其他组节点相连的节点称为边界节点,只与本组节点相连的节点称为内部节点。在高层次中,这些组被抽象成逻辑节点,并对这些逻辑节点进行类似地分组。这个过程可以递归地进行,形成层次结构,图 1 为文[4]中的层次结构。

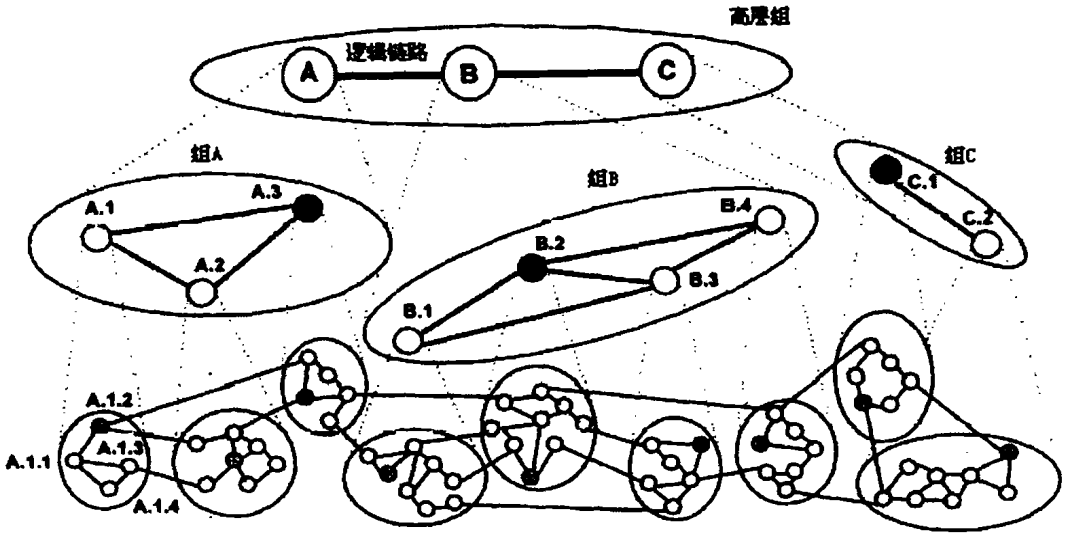

图 1 网络的层次结构

对网络进行层次结构划分后,低层的 QoS 信息只需要在本组扩散,不需要扩散到远处的组。高层并不知道低层内部详细的拓扑信息,只是将低层抽象为逻辑节点,因此降低了路由计算的复杂度,也减少了在网络中扩散的拓扑信息。路由算法要求高层对低层内部的拓扑信息进行压缩,否则会由于缺少必要的信息而无法计算出满足要求的路径。

2. 路由算法

拓扑压缩算法和路由算法有很大关系。如果路由算法只考虑跳数信息而不考虑其他 QoS 要求,则对拓扑压缩算法的依赖很弱。有多种 QoS 要求的路由算法,会因为 QoS 要求考虑的优先程度不同而采用不同的拓扑压缩算法。基于 QoS 的

路由选择算法大致可分成以下五类[6]：Widest shortest routing、Shortest widest routing、Competitive call routing、Shortest distance routing 和 Static minimum hop。

Widest shortest routing 是先在所有可行(feasible)路径中选出跳数最少的路径，在此基础上再选出带宽最大的路径，即优先考虑效率，让总带宽占用率最低。Shortest widest routing 正好相反，先选出带宽最大的路径，再选跳数最少的路径，即先考虑平衡负载，让各条路径上负载尽可能均衡。Competitive call routing 和 Shortest distance routing 均是定义一个与链路带宽相关的函数，然后选择函数值最小的路径，即尽量在效率和负载平衡间找到一个折中点。两者的区别是 Competitive call routing 中采用的是开根号函数，当可用带宽增加时，函数值减少得慢一些。Static minimum hop 则仅以跳数为度量，找出跳数最少的路径。

3. 拓扑压缩算法

经过拓扑压缩后，会损失一些信息，影响路由计算[15]。拓扑压缩算法的作用是用尽可能少的空间存储供路由计算使用的拓扑信息而不是所有的信息。实验证明[20]，在层次网络中根据压缩信息计算路由与在平面网络(非层次网络)中的性能(网络吞吐量和连接建立延迟)相近。在上层路由计算时，只需要知道下层分为哪几个组、各组之间的拓扑结构以及每组中任意两个边界节点之间的 QoS 信息。因此，拓扑压缩算法的目的就是用尽可能少的空间存储组中任意两个边界节点之间的路由计算所必需的 QoS 信息。

按保留的边界节点间的有用信息量和失真程度，可把拓扑压缩算法分成简单点算法、全相连算法、对称点算法、生成树算法和星型算法等 5 种。

3.1 简单点算法

简单点算法[2]丢弃组内节点间的所有信息，简单地将它压缩成一个逻辑点。这种算法适用于组内的信息对路由计算影响不大的情况。

3.2 全相连算法

全相连算法是去掉所有内部节点，只保留边界节点，在每一对边界节点之间构造一条虚链路，该虚链路的属性的值是根据实际网络拓扑信息计算出来的两个节点间所有链路的最优值。

该算法丢弃较差链路的信息，不影响路由计算；需要扩散的信息与边界节点的数目的平方成正比。当边界节点数远小于内部节点数时，该算法既达到压缩目的，又保留了最多的有用信息。但当边界节点与内部节点相当或远大于内部节点时，例如在环形网络中，每个节点都是边界节点，该算法既没有有效压缩信息，又损失了部分信息。极端情况下当各边界节点间的虚链路属性基本一样时，使用全相连算法甚至会增加冗余信息，与压缩相违背。

对于各种基于 QoS 的路由算法，全相连算法都能获到较低的 Call Failure Rate(Call Failure Rate = Total Number of Rejected Calls/Total Number of Requested Calls)，但是需要交换的信息最多。通常用它作为评价其他压缩算法的性能的参照。

3.3 对称点算法

对称点算法是将待压缩的组退化成一个点，赋予它一个属性，即直径。直径通常是压缩信息在组内的最差值或平均值，如最大带宽或平均时延。任意两个边界节点间虚链路的属性都取直径的值。扩散的信息就是直径。

该算法可实现信息的较大压缩；但当各边界节点间的虚链路属性差别较大时，该算法的信息失真十分严重。

当对称点算法应用在 Widest shortest routing 中时，性能明显比全相连算法差[9]。当应用于 Competitive call routing 和 Shortest distance routing 时，由于直径是所有路径的平均值或最差值，结果往往与所有路径中的最差值接近(即使在取平均值时也是如此[9])。因此当组内路径的 QoS 属性相差不大时，可以获得很好的性能。文[13]中也指出，对称点算法的性能取决于网络中边界节点间 QoS 属性的对称性。若各边界节点间 QoS 属性相差不大，则该算法的 Call Failure Rate 与全相连接近。

上述几种作法是极端的作法，简单点算法和对称点算法过于简化，损失了过多的有用信息，而全相连算法保留了过多的信息。为了在这几种算法之间找到一个平衡点，很多人已经做了大量有益的工作[1~3,7,14,17]，主要可有生成树(Spanning tree)算法和星型(Star)算法两种。

3.4 生成树算法

生成树算法是在全相连算法基础上找出一棵最小(大)代价生成树。对于带宽等凹性(concave)QoS 属性可以完全保留必要的信息从而恢复成与原来一样的全相连网络；对于时延等加性(addictive)QoS 属性将会损失部分必要信息，但能够恢复出原来全相连网络中 QoS 属性的上限。

· 凹性(concave)QoS 属性　以压缩组内带宽信息为例，凹性 QoS 属性的生成树算法包括两步：

1. 对待压缩组的每一对边界节点，计算最大带宽，构造边界节点间的全相连网络。

2. 从全相连网络中计算基于带宽的最大生成树。

可以证明全相连网络与最大生成树是等价的。即由最大生成树还原得到的全相连网络与计算最大生成树时输入的全相连网络是一致的[7]。

全相连网络中的需要存储的信息大小为 $O(n^2)$，而生成树中的需要存储的信息大小为 $O(n)$。但通过 $O(n)$的生成树可以毫无损失地还原成 $O(n^2)$的全相连网络。这是因为全相连网络中的各链路的带宽并不是相互独立，毫不相关的。任意两个节点间的最大带宽 d 必然不小于连接这两点的任意一条路径中链路带宽的最小值。当压缩有 N 个节点的全相连网络时，只需要保存 N－1 条链路的信息，即最多只有 N－1 条不同带宽的链路。通过找出生成树，可以将这些冗余信息去掉，达到既压缩信息又不损失信息的效果。

· 加性(addictive)QoS 属性　以压缩组内时延信息为例，加性 QoS 属性的生成树算法包括两步：

1. 对待压缩组的每一对边界节点，计算最小时延，构造边界节点间的全相连网络。

2. 从全相连网络中计算基于延迟的最大生成树。

根据该生成树也可以还原出时延的全相连网络，方法是：对于生成树上任意两个节点，若有链路直接相连，则该链路的时延即为全相连网络中该两个节点间的最小时延。若没有链路直接相连，则必然能在树上找到一条唯一的路径连接这两个节点。全相连网络中这两个节点间的最小时延就是这条路径上所有链路时延的最小值。对于时延的最大生成树中不直接相连的两个节点，虽然没有链路直接相连，但必有一条实线组成的路径连接这两点。设该路径上的实线链路时延分别为 a,b,c，还原后这两点虚链路的时延 d'，显然 d'=min{a,b,c}。

设原来全相连网络中连接这两点的链路时延为 d。由于该生成树是最大生成树，因此有 d<=min{a,b,c}；又因为 d 是两点之间的时延最小的路径，因此又有 d<=a+b+c，显然 d<=min{a,b,c}的条件更强，所以只能得出 d<=min{a,b,c}=d' 的结果，即 d' 是 d 的一个上界。d' 对 d 的逼近程度取决于实际网络中各链路的时延的范围。范围越小，则 d' 越逼近 d。设各链路最大时延为 d_{max}，最小时延为 d_{min}，显然 $d' <= d_{max}$，$d >= d_{min}$，因此有 $d' - d <= d_{max} - d <= d_{max} - d_{min}$。当 d_{max} 和 d_{min} 相差很小时，d' 和 d 相差也很小。

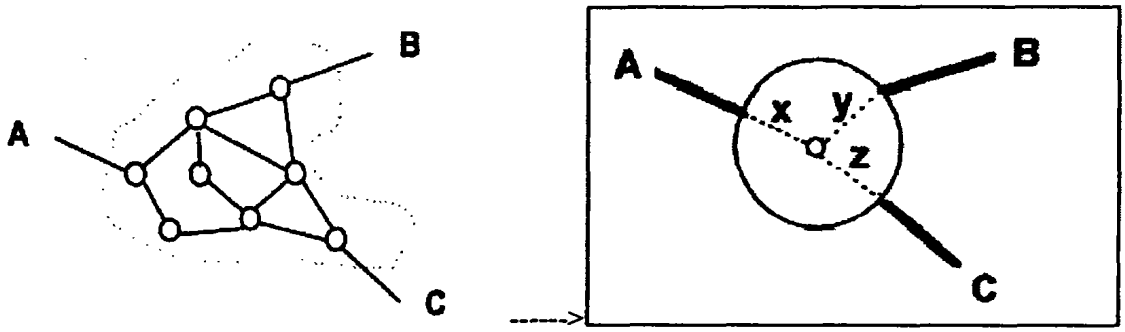

图 2　星型模型

星型算法的步骤有两步，首先构造与实际网络对应的边界节点全相连网络。然后计算出各边界节点到核心的虚链路 QoS 属性。在还原全相连网络中任意两个边界节点之间的 QoS 属性时，可以由这两个节点到核心的两条虚链路的 QoS 属性经过简单计算得到。问题的关键在于，如何计算各边界节点到核心的虚链路的 QoS 属性，使得根据它们还原出的边界节点间的虚链路 QoS 属性与真实情况相差尽可能小。

· 加性 QoS 属性　在星型算法中就是任意两个边界节点之间虚链路的 QoS 属性等于两个边界节点到核心的两条虚链路 QoS 属性之和。设有 N 个边界节点，A_{ij} 为真实网络中节点 i 和节点 j 之间的最优路径的 QoS 属性，X_i 为节点 i 到核心的虚链路 QoS 属性。问题转化为 X_i(i=1,2,…,N)如何取值使得 $F(X_1,X_2,\cdots,X_N)=\sum_{i<j}(X_i+X_j-A_{ij})^2$ 最小。用最小二乘法可得[16]：

$$X_i=\frac{(2N-3)\sum_{j=1}^{N}A_{ij}-\sum_{j=1}^{N}\sum_{k=1}^{N}A_{jk}}{2(N-1)(N-2)}\quad i=1,2,\cdots,N$$

易证 $X_i>0$。

· 凹性 QoS 属性　最小二乘法只能压缩加性 QoS 属性，星型算法对于带宽等凹性 QoS 属性则很难实现信息压缩。首先考虑，一个只有三个边界节点 A,B,C 的组。假设它们相互之间的最大带宽为 BW_{AB}，BW_{AC}，BW_{BC}，它们到核心的最大带宽分别为 BW_x，BW_y，BW_z。由前面生成树算法中的分析可知，BW_{AB}，BW_{AC}，BW_{BC} 中至少有两个相等，另一个不小于这两个。当 $BW_{AB}=BW_{AC}=BW_{BC}=BW$ 时，此时可将它们取为相同的值。当 $BW_{BC}>BW_{AB}=BW_{AC}$ 时，可取 $BW_y=BW_z=BW_{BC}>BW_{AB}=BW_x$。但是当推广到一般情况时问题则复杂得多。设有 N 个边界节点，它们到核心的虚链路带宽分别为 BW_i，则有

$BW_i>=\max\{BW_{ik}\}$ k=1,2,…,N

设 BW_{ab} 为全相连网络中 a,b 两个节点间虚链路的带宽，由于 $BW_a>=\max\{BW_{ak}\}$ k=1,2,…,N，$BW_b>=\max\{BW_{bk}\}$ k=1,2,…,N，又因为 $BW_{ab}=\min\{BW_a,BW_b\}$，所以有 $BW_{ab}=\max\{BW_{ak}\}$ k=1,2,…,N 或 $BW_{ab}=\max\{BW_{bk}\}$ k=1,2,…,N。若 BW_{ab} 是带宽最小的链路，则由上述结论推出 $BW_{ab}=BWak$ k=1,2,…,N 或 $BW_{ab}=BW_{bk}$，k=1,2,…,N。在实际网络中，各链路的带宽很难满足上述要求。因此，用生成树算法压缩凹性 QoS 属性，而用星型算法压缩加性 QoS 属性是一个不错的选择。

· 综合考虑两种属性　在星型算法中考虑两种属性时，可以在加性属性基础上增加对凹性 QoS 属性的比较，即在多条满足时延要求的路径中选择带宽最大或满足要求中带宽最小的路径。由于实际并不可能事先知道应用程序的时延和带宽要求，必须把组内多条不同时延和带宽的路径信息扩散出去以满足不同的应用程序的需求。因此在构造全相连网络时，需要考虑所有可能的路径，每条路径可以用一个二元组(时延，带宽)表示，即是平面上的一个点。多条路径可以有多个点表示。此时全相连网络中的虚链路包含的是多条路径的 QoS 信息，而不仅仅是一条路径的信息。UIUC 提出了一种算法用一条线段来近似多个点，从而达到压缩多条路径的信息的目的[3]。

星型算法应用于 Static minimum hop 时，明显比应用于其他考虑 QoS 请求的路由算法时的性能差[5]，当网络负载高的时候，Static minimum hop 中的 Call Failure Rate 比 Competitive call routing 高 60%。但是当同时应用于 Static minimum hop 时，星型算法与全相连算法的性能没有大的区别，它们都能将边界节点间的精确的或近似精确的跳数信息扩散出去。

当生成树算法应用于 Static minimum hop 时，性能比全相连和星型稍差一些。试验证明生成树算法的 Call Failure Rate 比全相连算法高 15%[5]，这是因为生成树算法无法对跳数(加性 QoS 属性)进行无损压缩。

3.5　星型算法

星型算法(又叫 Complex node representation[7])采用 ATM PNNI 中的复合节点模型[12]，每个组都保留所有边界节点，并将内部节点抽象为一个逻辑节点(即核心)，所有边界节点都有一条虚链路与核心相连，如图 2 所示[4]。

4. 性能比较

拓扑压缩算法的性能除了和算法本身有关以外，还收到很多其他因素的影响，例如拓扑结构，更新策略等。同样的压缩算法在不同的拓扑结构或更新策略下的性能可能相差很大。

4.1　拓扑结构的影响

当网络节点较少时，可以把整个网络抽象成一个平面图，每个节点都了解整个网络的拓扑结构，并据此进行路由计算。当网络节点增多时，为了解决由于节点数增加而导致的路由计算量增加，产生了层次网络结构[19]。层次网络拓扑结构大致可分成 edge-core 和 n-clusters 两类[8]。在 edge-core 拓扑结构中，由若干个骨干路由器(core router)构成一个骨干传输网络，主要负责传输数据。它们之间的传输链路通常带宽较大。各个组通常有若干个边界路由器(edge router)和骨干网中的骨干路由器通过高速链路相连，但是各个组之间并无链路直接相连。组内其他路由器和边界路由器通过低速链路相连。n-clusters 拓扑结构是将所有路由器分成多个组，每个组有若干边界路由器与其他组的边界路由器通过高速链路相连，组内的路由器间则是通过低速链路相连。

在 edge-core(也叫 transit-stub[22])拓扑结构中，由于组间没有链路直接相连，任意一个跨组的连接都是从源组通过骨干传输网络到达目的组的，因此只需将骨干网的拓扑信息扩散到所有组即可。源组不用关心目的组的拓扑结构，甚至也不用关心目的组压缩后的拓扑信息，只需要知道骨干网的拓扑信息就能计算出路径。因此此时采用简单点算法，把各个组压缩成骨干网中的逻辑点，可以获得最大的"性价比"。文[8]中模拟试验证实了该结果，此时简单点算法和全相连算法有相同的 Call Failure Rate。

在 n-clusters 网络中，除了叶子组，其他组都有可能成为传输组，为某个连接起传输作用，因此此时该组的拓扑信息对路由选择有一定作用。所以简单点算法不适合这类拓扑结构。文[8]中试验证明此时全相连算法的 Call Failure Rate 最小，但是压缩后的信息在扩散时需要耗费的带宽最大。而简单点算法的 Call Failure Rate 最大。其他几种算法都不同程度地保留了组内拓扑信息，可以在路由计算时利用。例如，若在层次结构中各边界路由器间虚链路的 QoS 属性相差不大，则采用对称点算法比较好。在大部分情况下，星型算法和生成树算法能够在不影响路由性能的基础上在 Call Failure Rate 和耗费带宽上找到一个平衡点。

网络传输瓶颈的位置对层次结构划分和路由计算结果有一定影响。在 n-clusters 拓扑结构中，若组间的带宽与组内的带宽接近，则瓶颈出现在组间的链路，由于最优路径往往取决于瓶颈，计算出的组间路径可以接近最优路径。若组间的带宽远大于组内带宽，瓶颈出现在组内，如果组内的压缩信息不能很好反映瓶颈的影响，则计算出的路径可能与最优路径有较大偏差。因此，层次结构划分应尽量使得瓶颈位置的信息能在压缩信息中体现出来，这样有利于拓扑压缩和路由选择取长补短，提高整体性能。

拓扑结构的连通度和直径(此处直径表示网络中任意两点之间的最大跳数)对压缩算法的性能也有影响。文[11]中对于规则的 k-ray n-cube 网络进行了模拟试验。其中 n 越大表明连通度越大，k 越大表面直径越大。对于连通度大的网络，由于两点间有很多可供选择的链路，因此 Call Failure Rate 的几率也更低。但是由于连通度大，在组内需要扩散的链路信息也更多，一定程度抵消了链路选择余地大带来的好处。随着直径的增加，边界节点间的路径包含的链路也就越多，但是压缩后只保留一条虚链路的信息，因此损失的信息也越多。此时路由程序根据不准确的压缩信息进行路由选择产生 crankback(crankback 指在路由过程中没有满足要求的路径而产生的回溯)的几率也越大。文[5]中指出在 staged ring 拓扑结构中，crankback 与直径成线性关系。

4.2 更新策略的影响

链路的 QoS 信息是不断地变化的，必须对这些信息及时更新使得路由计算符合当时的网络状态。信息更新不能太快，否则会增加网络负载，得不偿失；而更新太慢，会使路由计算因为过时信息而作出错误的路由选择。更新策略是 QoS 路由中必须考虑的难题。

更新策略可分为两类：定时更新和触发更新。

定时更新是设定一个定时器，超时时更新。触发更新是指当有重大变化才进行更新。触发更新策略中通常还有一个定时器，用来约束两次更新间的最小时间间隔。当网络 QoS 信息变化频繁的时候可能导致频繁的触发更新使得网络性能降低，设定一个最小更新间隔定时器(clamp down timer)，只有在它超时以后才能进行下一次的更新。

触发更新可分为两类：阈值更新和区间更新。

阈值更新是指事先指定一个固定的阈值 th，设上一次收到的 QoS 值为 Q0，当前收到的 QoS 值为 Q1，若 $(Q1-Q0)/Q0>th$，则触发更新。

区间更新是指将可能的 QoS 值的区间分为多个连续的小区间，当当前的 QoS 值与上一次的 QoS 值不在同一个小区间时触发更新。按照小区间的划分方法又可分为等值更新和指数更新。在等值更新中，可能的 QoS 值的区间被分为大小相同的多个小区间：(0,B)，(B,2B)，(2B,3B)，…。在指数更新中，QoS 值的区间分为多个大小呈指数增加的小区间，此时的小区间是(0,B)，(B,(f+1)B)，((f+1)B,(f^2+f+1)B)，…，B 和 f 都是常量。

区间更新的一个缺点是当 QoS 值在区间两端附近小范围变化会导致频繁的更新。可以可采取一种滞后机制避免这种情况[9]。

通常更新策略都有一个最小更新间隔定时器，用于防止网络 QoS 信息变化频繁导致过多的更新信息而降低网络性能。最小更新间隔的选取对更新分组的数量起决定作用。当最小更新间隔过大时，各更新策略产生更新分组的速率达到一个定值，且相差不大，与更新策略无关。当最小更新间隔很小时，区间更新策略产生更新分组的速率随最小更新间隔增大而迅速减小，B 值越小的变化越明显；定时更新则按照定时器值的大小分为两种情况：定时时间长时，产生更新分组的速率不受最小更新间隔的影响，速率保持恒定；定时时间短时，产生更新分组的速率随最小更新间隔增大而迅速减小[10]。在相同最小更新间隔情况下，B 值越大的区间更新产生更新分组的速率越小；定时时间越长的定时更新产生更新分组的速率越小。

更新策略对压缩算法的性能也有影响。若采用定时更新策略，且更新周期较长时，对称点算法和全相连算法的性能相差不大，甚至优于全相连算法，这与直觉恰好相反[9]。这是因为，全相连算法不能将当前信息及时扩散出去，影响路由计算的正确性；而对称点算法扩散的组直径变化远小于各虚链路属性的变化，当各虚链路属性的变化是独立随机事件时过时的直径与当前直径相差不大。

更新策略参数的选取也会影响路由算法的性能。在定时更新策略中，若定时器设置的时间过长，基于 QoS 的路由算法的性能可能比静态路由还要糟糕[11]。这是因为路由器没有获得准确的拓扑信息而误将已经不满足要求的路径作为满足要求的路径或刚好相反，从而导致 Call Failure Rate 增加。在

触发更新中，各种触发条件的参数对网络负载和路由算法性能的影响也不可忽视。随着 th 和 B 的减小，更新策略变得更加"敏感"，这样无疑会增加信息更新分组从而增加网络负载，但同时也会提高路由算法的性能，因为此时路由算法获得的链路状态信息与当前链路状态信息相同或相近。为了在不增加网络负载的前提下尽量提高路由算法的效率，可以针对更新策略所带来的不确定性，在路由算法中引入概率的思想。例如在触发更新策略中，根据每条链路最近更新的 QoS 信息和当前的 QoS 要求为它们分配一个 safety 值，代表它们能够满足该要求的概率，再根据此 safety 值进行路由选择[21]。为了弥补更新周期过长带来的信息过时的缺点，可采取 Randomized routing[10]，随机地从多条满足最低要求的路径中找出一条路径，这样可以有效地减少 Call Failure Rate。

结论与今后的工作 本文讨论了几种已有的层次网络中的拓扑压缩算法，通过对它们在不同拓扑结构和更新策略下性能的比较，可以看出大部分情况下全相连算法具有最优的性能，但是在某些特定的条件下，其他压缩算法可以获得与全相连算法相近甚至更优的性能。例如在 edge-core 网络中对 edge 网采用简单点算法，在更新周期长的情况下采用对称点算法。但是全相连算法需要扩散的信息过多，星型算法和生成树算法不但能有效地减少需要扩散的信息量，并且在性能上与全相连网络相近。

影响压缩算法性能的因素有很多，有拓扑结构的划分方法，更新策略的选取和路由算法等等。此外，网络负载的分布对压缩算法也有影响，直观地想，在网络负载比较均匀的情况下，采取对称点算法应有不错的性价比。在路由算法中如何设计一个带宽的函数，使得 Call Failure Rate 最小。在区间更新策略中，等值更新与指数更新各适用于哪些情况？这些都是今后的研究中需要考虑的问题。

参 考 文 献

1 Lee W C. Spanning Tree Method for Link State Aggregation in Large Communication Networks. IEEE INFOCOM, 1995. 297～302

2 Private network-network interface specification version 1.0 (PNNI): [Technical report]. The ATM Forum technical committee, March 1996, af-pnni-0055.000

3 Lui K S, Nahrstedt K. Heirarchical QoS Routing in Delay-Bandwidth Sensitive Networks. In: Global Telecommunications Conf. 2000, GLOBECOM '00. IEEE , Volume: 1 , 2000. 410～414

4 Van Mieghem P. Topology information condensation in hierarchical networks. Computer Networks ,1999,31:2115～2137

5 Awerbuch B, Du Yi, Khan B. Routing Through Networks with Hierarchical Topology Aggregation. Computers and Communications, 1998, ISCC '98 Proc. Third IEEE Symposium on , 1998. 406～412

6 Hao Fang, Zegura E W. On Scalable QoS Routing: Performance Evaluation of Topology Aggregation. IEEE INFOCOM 2000. 147

7 Lee W C. Topology aggregation for hierarchical routing in ATM networks. ACM Sigcomm, 1995, 25(2): 82～92

8 Dhandapani G. A Performance Evaluation Architecture for Hierarchical PNNI and Performance Evaluation of Different Aggregation Algorithms in Large ATM Networks: [Technical Report]

9 Hao F, Zegura E W. Scalability Techniques in QoS Routing. Networking and Telecommunications Group, College of Computing Georgia Tech, Atlanta

10 Apostolopoulos G, Guerin R, Tripathi S K. Quality of Service Based Routing: A Performance Perspective. In: Proc. of ACM SIGCOMM'98 Conf. Sep. 1998

11 Shaikh A, Rexford J, Shin Kang G. Dynamics of Quality of Service Routing with Inaccurate Link State Information: [Technical Report CSE-TR-350-97]. Computer Science and Engineering Division, Department of Electrical Engineering and Computer Science, Universirty of Michigan, Nov. 1997

12 Guo L, Matta I. On State Aggregation for Scalable QoS Routing. ATM Workshop Proceedings, 1998 IEEE , 1998. 306～314

13 Hao F, Zegura E W. Bhatt S. Performance of the PNNI Protocol in Large Networks. ATM Workshop Proceedings, 1998 IEEE , 1998. 315～323

14 Awerbush B, Shavitt Y. Topology Aggregation for Directed Graph. In: Third IEEE Symposium on Computers and Communications, June 1998

15 Guerin R, Orda A. QoS Based Routing in Networks with Inaccurate Information: Theory and Algorithm. In: Proceedings of INFOCOM, 1997

16 刘进. 大规模层次化网络中保证服务质量的路由算法: [博士学位论文]. 清华大学电子工程系, 2000

17 Iwata A, Suzuki H. QoS Aggregation Algorithms in Hierarchical ATM Networks. Communications, 1998. ICC 98. Conf. Record. 1998 IEEE Intl. Conf. on , Volume: 1, 1998. 243～248

18 Van Mieghem P. Dividing A Network into Peer Groups to Build A Hierarchical Structure

19 Tangmunarunkit H, Govindan R. Network Topologies, Power Laws, and Hierarchy: [Technical Report, ISI]. University of Southern California

20 Awerbush B, Du Yi, Shavitt Y. The Effect of Network Hierarchy Structure on Performance of ATM PNNI Hierarchical Routing. In: Proc. of the Intl. Conf. on Computer Communications and Networks, 1998

21 Apostolopoulos G, Guerin R. Improving QoS Routing Performance Under Inaccurate Link State Information. In: Proc. of ITC'16, Edinburgh, Scotland, June 1999

22 Zegura E W, Calvert K L. How to Model an Internetwork, INFOCOM '96, Fifteenth Annual Joint Conf. of the IEEE Computer Societies. Networking the Next Generation, Proceedings IEEE , Volume: 2 , 1996. 594～602

（上接第 52 页）

合理地使用处理、缓存等资源。

结论 本文针对 IP 网络的实时多媒体应用阐述了在传输两端进行非均匀 QoS 控制的思想及其基于 RTP 协议和 MPEG-4 扩展编码的实现方法。传输前的非均匀带宽资源分配使得带宽得以更加合理有效的利用。通过对 RTP 包头进行重新定义，使得接收端能够快速区分基本层包和增强层包，并准确判断各层的丢包情况和进行非均匀的传输错误处理。通过对部分 RTP 包头的扩展，又使得接收端对传输延迟变化的监测更为准确和及时。非均匀 QoS 控制能够保证流媒体应用的自适应性和实时性，及更合理地使用带宽、处理、缓存等资源。

参 考 文 献

1 Braden B, et al. Recommendations on Queue Management and Congestion Avoidance in the Internet. RFC 2309, April 1998

2 Wu Dapeng, et al. Streaming Video over the Internet: Approaches and Directions

3 ISO/IEC 14496: MPEG-4, Oct. 2000

4 Schulzrinne H, et al. RTP: a transport protocol for real-time application. Internet Engineering Task Force, RFC 1889, Jan. 1996

5 Handley J. SDP: Session Description Protocol. RFC 2327, Internet Engineering Task Force, April 1998

6 Braden R, Ed. Zhang L, Berson S, Herzog S, Jamin S. Resource ReSer Vation Protocol(RSVP). Sep. 1997

7 DARPA Internet Protocol. RFC 791, Sep. 1981