在接收端模拟 TCP 的组播拥塞控制协议*)

原 冰 张 冰 刘增基
(西安电子科技大学综合业务网国家重点实验室 西安 710071)

摘 要 提出了一种在接收端模拟 TCP 行为的组播拥塞控制协议 TEARM。该协议在每个接收端独立地维护拥塞窗口并模拟 TCP 协议来改变窗口大小,其后将窗口值转换为期望速率,反馈给发送端,其中反馈的速率为一段时间内的加权平均。此外,使用了基于代表的机制来抑制反馈,并使用了历史打折机制来提高协议的响应性。仿真表明,该协议具有良好的 TCP 公平性、速率平滑性、可扩展性以及响应性,适用于流媒体组播业务的传输。

关键词 拥塞控制,组播,TCP 公平性,可扩展性

TCP Emulation at Receivers for Multicast Congestion Control

YUAN Bing ZHANG Bing LIU Zeng-Ji
(State Key Lab. of Integrated Services Networks, Xidian University, Xi'an 710071)

Abstract TCP emulation at receivers for multicast congestion control protocol is proposed in the paper. The protocol maintains an independent congestion window at each receiver and imitates TCP to change the window size, which is then converted into the expected rate that is a weighted average during a period of time, and then the rate is fed back to the sender. Besides, the mechanism based on the representative is used to suppress the feedbacks and the history discount mechanism is introduced to improve the responsiveness of the protocol. The simulations indicate that the protocol shows good TCP-fairness, smoothness, scalability and responsiveness, which makes it a good choice for streaming media transportation.

Keywords Congestion control, Multicast, TCP-friendliness, Scalability

在当前的网络设计中,组播拥塞控制所要解决的问题比单播拥塞控制更为复杂[1]。这是由于组播接收端数目众多,而且接收端可能经历不同的网络环境,此外各个接收端可能随时加入或退出组播组,这些都对组播拥塞控制的设计带来很大的难度。针对组播自身的特点,出现了基于窗口和基于速率这两类主要的组播拥塞控制协议。前者主要代表为 MTCP (Multicast TCP congestion control)[2],后者主要代表为 TFMCC (TCP-Friendly Multicast Congestion Control)[3]。衡量组播拥塞控制协议性能的主要方面是 TCP 公平性[4]和可扩展性。对于可扩展性来说,在组播中还需要解决反馈内爆问题[5]和丢失路径多样性[6]的问题。MTCP 和 TFMCC 两个协议都能达到良好的 TCP 公平性,但是 TFMCC 由于受限于模型以及参数测量的准确性,MTCP 实现的复杂性都严重限制了两者的可扩展性。在保证对 TCP 公平的前提下,如何提高机制的可扩展性成为了组播拥塞控制研究中的核心问题。

在单播拥塞控制机制中,TEAR[7] (TCP Emulation At Receiver)协议具有良好的 TCP 公平性、平滑性和响应性。我们在 TEAR 的基础上,提出了实现组播拥塞控制的 TEARM。TEARM 在接收端模拟了 TCP 在发送端的控制机制,在每个接收端都根据网络拥塞状况来调整拥塞窗口,并将窗口转化为对 TCP 公平的发送速率,这样既保证了协议的 TCP 公平性,又使得协议实现非常简单;TEARM 采用在一段较长时间内对瞬时速率进行加权平均的方法,估计期望速率,避免了发送速率的锯齿型抖动,使得速率变化更为平滑;由于在接收端进行拥塞检测和拥塞控制,组播组规模的扩大并不会给 TEARM 的发送端带来更多的负担,大大改善了协议的可扩展性;TEARM 采用基于代表的机制,只有被发送端选为代表的接收端才能定期反馈速率,很大程度上降低了各接收端的反馈信息量,有效避免了反馈内爆的问题;每个接收端独立地维持拥塞窗口,解决了丢失路径多样性的问题;TEARM 的历史打折机制进一步提高了协议的响应性。

本文在详细描述了 TEARM 机制后,在 NS-2[8]平台下对 TEARM 做了较为全面的仿真,并与 TFMCC 进行了比较。通过对仿真结果的分析,可看出 TEARM 与 TFMCC 相比具有良好的 TCP 公平性、平滑性和可扩展性。这些特性使得 TEARM 适用于流媒体组播业务的传输。

1 TEARM 的协议描述

TEARM 协议由发送端和接收端两部分功能组成。

1.1 发送端功能

发送端在接收端中选择期望速率最小的接收端作为代表,并根据代表的反馈信息调整发送速率。同时,发送端还需要协助接收端测量往返时延以及控制接收端的反馈。

1.1.1 选择代表

发送端从所有组播组的接收端中选取期望速率最小的接收端作为当前受限接收端(Current Limited Receiver, CLR)。CLR 作为组播组接收端的代表周期向发送端反馈其估计速率。如果其他接收端的反馈速率低于 CLR 的速率,则更新 CLR;若 CLR 离开,或是在连续 10 个最大往返时延(MAX_

*)国家自然科学基金项目资助(90104012)。原 冰 在读硕士生。

RTT)内没有收到来自 CLR 的反馈,则发送端认为该接收端已经离开组播组,立即重新选择 CLR。

1.1.2 调整发送速率

发送端根据 CLR 的反馈信息,即估计速率来调整发送速率。当在 15 个 MAX_RTT 的时间内没有收到任何来自接收端的反馈信息,发送端认为链路严重拥塞,则将发送速率减半来适应网络的拥塞状况。

1.1.3 协助接收端测量往返时延(RTT)

发送端还要协助各接收端测量 RTT。各接收端在反馈包中包含该反馈包发送的时刻,发送端收到来自不同接收端的反馈包后,按优先级选择一个接收端,并协助其测量 RTT。方法是发送端在随后发送的数据包中,填入该接收端的 ID,并记录该接收端反馈包到达发送端的时刻以及发送当前数据包的时刻。该接收端在收到含有自身 ID 的数据包时,即可计算出 RTT;计算时要扣除发送端接收到反馈和发送数据包的间隔时间。具体公式如下:

$$RTT=\text{包到达接收端的时刻}-\text{接收端发送反馈的时刻}-Offest \tag{1}$$

$$Offset=\text{发送端发送包的时刻}-\text{反馈到达发送端的时刻} \tag{2}$$

在协助接收端测量 RTT 时,发送端选择接收端的优先级是:没有测量过 RTT 的 CLR,其优先级最高,其次为没有测量过 RTT 的非 CLR 接收端,接着是测量过 RTT 的非 CLR 接收端,最后是测量过 CLR 的接收端。

1.1.4 控制接收端反馈

如何有效地避免反馈内爆是协议需要解决的问题。这里,引入 TFMCC 中的反馈轮机制。在反馈轮(反馈轮的间隔为 RTT_max 的 6 倍)中,发送端在发送的数据包中包含 RTT_max,非 CLR 的接收端只有 RTT 值高于 RTT_max 时也可以发送反馈。而 CLR 是不受反馈轮控制的,并周期地向发送端发送反馈。

1.2 接收端功能

在每个接收端都独立维持拥塞窗口,并在接收端模拟 TCP 行为,根据拥塞信息来调整窗口,将窗口值转化为期望速率,并对期望速率进行一段时间内的加权平均。引入历史打折机制来提高 TEARM 对于网络环境变化时的响应性。同时,每个接收端在反馈定时器溢出时,向发送端发送包含期望速率的反馈。

1.2.1 轮

在 TEARM 中引入"轮"的概念来模拟 TCP 的拥塞窗口调整机制。接收端每一轮结束时,调整拥塞窗口并计算估计速率,并且每轮结束时向发送端发送反馈。一般情况下,若当前拥塞窗口为 *cwnd*,则将接收端每收到 *cwnd* 个数据包的过程称为一轮。每收到 *cwnd* 个包,即认为当前轮结束,并开始新的一轮。当接收端检测到拥塞发生时,即接收端发现丢包或者超时时,则当前轮立即结束,并在等待一个 RTT 后,重新开始新一轮。

1.2.2 状态转移

TEARM 协议在各接收端模拟了 TCP 的拥塞控制机制,因此相应设置了 4 个状态:慢启动、拥塞避免、快速恢复和超时。

在最初的时候,当接收端收到一个数据包后,自动进入慢启动状态。当 cwnd 超过慢启动阈值 ssThresh 时,从慢启动状态进入拥塞避免状态。与 TCP 相同,TEARM 也是用丢包作为拥塞指示。在慢启动状态或是拥塞避免状态,如果遇到丢包,则认为网络发生了拥塞,进入快速恢复状态,并等待一个 RTT 后进入拥塞避免状态。在此 RTT 内忽略所有收到的包,这是为了模拟 TCP 在同一丢失事件中所采取的机制。如果接收定时器(其时长为 4 * RTT)超时,则进入超时状态,同样等待一个 RTT 后,进入慢启动状态,并将 ssThresh 设置为 min[*cwnd*,2]的一半。

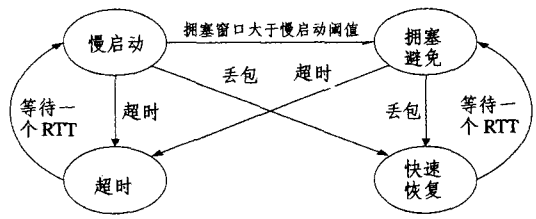

图 1 TEARM 的状态转移图

在接收端保存着接收数据包的历史记录,并据此来判断网络是否发生丢包。在慢启动状态或是拥塞避免状态,接收端收到不按序到达的包,TEARM 就假定应该按序到达的那个包在网络中丢失了。如果累计收到 3 个序号大于假定丢失包序号的数据包,则认为那个包确实已经丢失,接收端进入快速恢复状态。如果在收到不按序到达的包后,收到了之前假定认为丢失的包,这是由于网络中的包乱序造成的,那么将返回之前的慢启动或是拥塞避免状态,并且更新 cwnd。这里模拟了 TCP 的 3 个重复 ACK 触发快速恢复的机制。

1.2.3 窗口调整

·增加策略

在 TEARM 的接收端,每轮结束时,如果没有发生丢包或者超时,那么增加接收端的窗口值 *cwnd*。为了模拟 TCP 在慢启动状态每收到一个 ACK,*cwnd* 就加 1。也就是说,每收到 *cwnd* 个包,*cwnd* 就翻倍。TEARM 在慢启动状态,每轮结束时 *cwnd* 翻倍,见公式(3)。类似地,TCP 在拥塞避免状态,每收到一个 ACK,*cwnd* 就增加 1/*cwnd*。也就是说,每收到 *cwnd* 个包,*cwnd* 就加 1。TEARM 在拥塞避免状态,每轮结束时 *cwnd* 以 1 的幅度增加。

$$\begin{aligned} &SS:\ cwnd\leftarrow cwnd*2\\ &CA:\ cwnd\leftarrow cwnd+1 \end{aligned} \tag{3}$$

·减小策略

同样,为了模拟 TCP 在进入快速恢复状态后发送端将 *cwnd* 减半的策略,TEARM 的接收端在进入快速恢复状态后将 *cwnd* 减半,见公式(4)。TCP 在超时后,在进入超时状态后,*cwnd* 设置为 1,而且将超时定时器的时间翻倍。

$$\begin{aligned} &FF:\ cwnd\leftarrow cwnd/2\\ &TO:\ cwnd\leftarrow 1 \end{aligned} \tag{4}$$

1.2.4 速率计算

TEARM 借鉴了 TEAR 中 epoch 的概念。TEARM 定义 epoch 为一个"锯齿"周期。一个 epoch 开始于当接收端进入慢启动状态或者是拥塞避免状态,结束于发生包丢失后,接收端再次进入慢启动状态或是拥塞避免状态。一个 epoch 模拟了 TCP 经历窗口的加性增加和乘性减少(AIMD)这一过程的时间。一个 epoch 可能包含一个或多个轮。TEARM 对 W 个 epoch 中的平均速率做加权平均,得到稳定的长期吞吐量,这样避免了 TCP 的瞬时速率震荡。

在一个 epoch 中,TEARM 接收端记录多个轮的 *cwnd* 和 *RTT* 或是 *RTO* 值。假设当前为第 k 个 epoch。在每一轮结

束的时候，接收端用在该 epoch 中记录的 RTT 或是 RTO（注意一个 epoch 中只能有一个 RTO）的总和去除在第 k 个 epoch 中记录的 $cwnd$ 样本总和。得到的结果就是第 k 个 epoch 的速率样本。

在每轮的末尾，如果第 k 个 epoch 没有结束，其速率样本值不能反映该 epoch 的平均速率，所以 TEARM 接收端计算两个加权平均的 epoch 速率：一是对第 $k-1$ 个 epoch 到第 $k-(W-1)$ 个 epoch 样本做加权平均，另一是从第 k 个 epoch 到第 $k-(W-1)$ 个 epoch 的加权平均。从两个平均值中选择较大的一个乘以包大小，作为向发端反馈的反馈速率，见公式(5)。如果速率样本少于 W 个 epoch（也就是 $k<W$），那么速率样本设置为 0。对于当前的设置，我们选择 W 为 8。

$$rate=psize*\max\left[\frac{1}{W-1}*\sum_{k-1}^{k-(W-1)}w_i*rate_i,\frac{1}{W}\sum_{k}^{k-(W-1)}w_i*rate_i\right] \quad (5)$$

我们使用表 1 所示的权重来平滑 epoch 速率。

表 1　W 个 epoch 的权值

epoch	k	k−1	k−2	k−3	k−4	k−5	k−6	k−7
weight	1.0	1.0	1.0	1.0	0.8	0.6	0.4	0.2

1.2.5　历史打折

为了使 TEARM 在网络拥塞状况迅速改善的时候提高响应性，应当尽可能地减小过去的速率样本所占的权重。这便是历史打折算法的思想。历史打折算法首先为每一个速率样本赋一个动态变化的权值 DF_i 和一个综合打折因子 GDF，它们的初值都为 1。在每轮结束时，如果当前轮计算的速率大于已记录的各 epoch 的平均速率的 a（协议中 a 取 1.4）倍以上，就认为网络从拥塞中恢复或是网络带宽突然增加，则更新 GDF 为二者的比率。

在每个 epoch 的末尾，DF_i 进行如下更新。

(1)对历史因子进行综合打折：

$$DF_i=GDF*DF_i \quad (6)$$

(2)将折扣因子随丢失间隔切换：

$$DF_{i+1}=DF_i \quad (7)$$

(3)初始化：

$$DF_0=1,DF=1 \quad (8)$$

在计算速率的时候，采用如下公式计算速率：

· 不包含当前 epoch 的计算速率

$$w=\sum_{k-1}^{k-(W-1)}w_i*DF_i*GDF$$

$$rate=\frac{1}{w}\sum_{k-1}^{k-(W-1)}rate_i*w_i*DF_i*GDF \quad (9)$$

· 包含当前 epoch 的计算速率

$$w=\sum_{k}^{k-(W-1)}w_i*DF_{i-1}*GDF$$

$$rate=\frac{1}{w}\sum_{k}^{k-(W-1)}rate_i*w_i*DF_{i-1}*GDF \quad (10)$$

由上式可以看出，GDF 和 DF_i 都减小了过去样本的权重，增大了最近的估计速率的权重，有效地减小了过去对现在的影响，使 TEARM 能够快速地从拥塞恢复到合适带宽。

1.2.6　反馈抑制

被选择为 CLR 的接收端，每一轮发送一次反馈，其他接收端设置反馈定时器来抑制反馈。CLR 独立于其他的接收端，每一轮发送反馈。CLR 的反馈不抑制其他接收端的反馈。非 CLR 接收端在反馈轮开始时设置反馈定时器。使用 TFMCC 中的随机定时器。TEARM 按(11)式设置反馈定时器：

$$t=\max(T*(1+\log(x)/\log(N)),0) \quad (11)$$

上式中，x 是服从 (0,1] 间均匀分布的随机变量，T 是反馈轮的间隔，N 是接收者数目上限的估计值。协议中推荐把 N 设置成 10,000。接收端在等待反馈定时器超时后，才能发送反馈包，除非在此之前就取消了定时器。如果数据包中的抑制速率高于接收端计算的期望速率，或是 RTT_max 大于等于接收端测量的 RTT，取消定时器。同样地，数据包中包含新的反馈轮开始的指示，则取消所有对旧轮的反馈。这样，有效地抑制了非 CLR 接收端的大量反馈。

2　TEARM 协议性能仿真

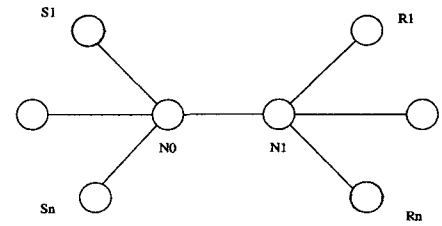

图 2　哑铃型拓扑结构

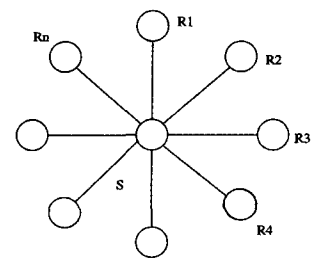

图 3　星型拓扑结构

仿真在 NS-2 平台上进行。本文中的仿真或者采用图 2 的哑铃型拓扑或者是图 3 的星型拓扑。包大小为 1000 字节，TCP 采用 Reno 版本。

2.1　公平性和平滑性

采用哑铃型拓扑。N0 和 N1 之间的瓶颈链路时延为 50ms。瓶颈链路设置为 RED 方式，队列大小为两倍的带宽时延积，minThresh 为 0.2 倍的队列大小，maxThresh 为 0.8 倍的队列大小。6 条 TEARM 流和 6 条 TCP 流竞争 12Mbps 的瓶颈带宽，每个 TEARM 流有 15 个接收端。为进行比较，对 TFMCC 协议也进行了同样的仿真。

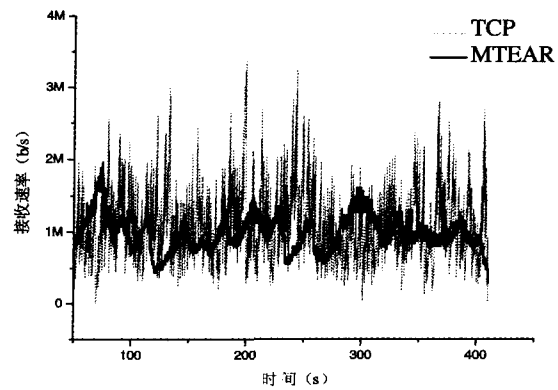

图 4　TEARM 流与 TCP 流的瞬态速率

由图 4 可以直观地看出，TEARM 的接收速率基本稳定在 1Mbps 左右，而 1Mbps 是 6 条 TEARM 和 6 条 TCP 在 12M 链路上竞争的公平速率，所以 TEARM 能够达到对 TCP 的公平。图 5 为在不同时间尺度下，TEARM 和 TFMCC 对 TCP 的等价比 EQ(EQ 和下文 COV 的定义详见文[9])。图中 EQ 值体现了不同时间尺度下与 TCP 的公平性。从对 EQ 值的分析可以得出，TEARM 的 EQ 曲线略高于 TFMCC，说明 TEARM 和 TFMCC 的 TCP 公平性基本相当。

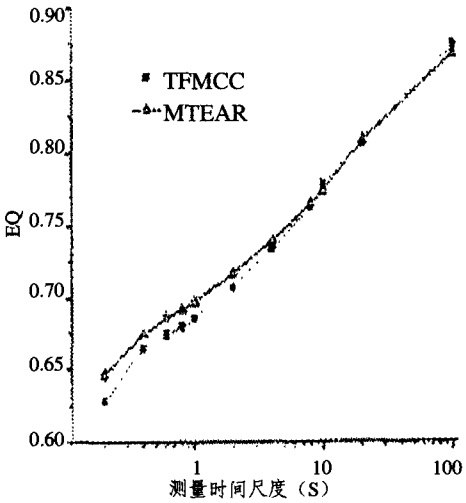

图 5　TFMCC 和 TEARM 的等价比 EQ

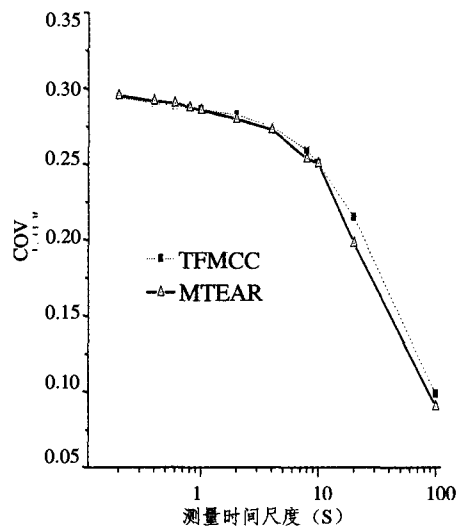

图 6　TFMCC 和 TEARM 的偏差系数 COV

由图 4 可以看出，TCP 在 0M 和 3M 之间剧烈抖动，而 TEARM 在 1.5M 和 2M 之间波动，TEARM 速率抖动的范围远比 TCP 小。从图 6 对不同时间尺度下偏差系数 COV 的分析上来看，TFMCC 的 COV 曲线和 TEARM 的曲线几乎重合，说明二者具有相同的速率平滑性。平滑的速率能够避免视频数据的振荡，因此 TEARM 更适用于流媒体业务的传输。

2.2　可扩展性

采用哑铃型拓扑。瓶颈链路带宽为 8Mbps，时延为 50ms，队列采用 DropTail 方式，队列大小为 150。接收端数目按照 2 的指数次幂增加，1，2，4，…，512。记录每次仿真的平均吞吐量(即发端的发送速率)以及 Ack 的数目与数据包数目比率的平均值。

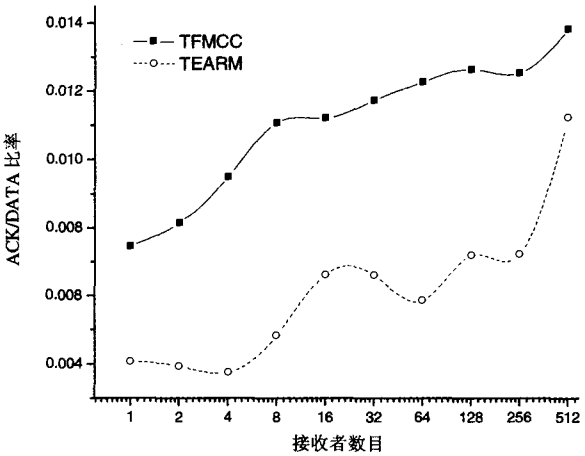

图 8　TEARM 和 TFMCC Ack 数目和数据包数目的比率

随着接收端数目的增加，TEARM 和 TFMCC 的发送速率一直很稳定。另一方面，从图 8 可见，虽然 TEARM 的 ACK 与数据的比率随着接收者数目的增加而不断增大，但是却一直低于在同样仿真环境中 TFMCC 的比率。TEARM 成功地解决了反馈内爆问题。由此看出，随着接收端数目的增加，TEARM 在吞吐量并没有出现明显下降的情况下，反馈抑制机制也将 ACK 的数目控制在可以接受的范围内。所以说 TEARM 具有良好的可扩展性。

2.3　响应性

2.3.1　对链路丢失率的响应性

采用星型的拓扑结构。TEARM 有 4 个接收端，分别设置 4 个接收端的链路丢失率为 0.1%，0.5%，2.5% 和 12.5%。在 0s 时丢失率为 0.1% 的接收端加入。从 200s 开始，每 100s 按丢失率从小到大依次加入。在 500s 时，接收端按丢失率从大到小依次退出。

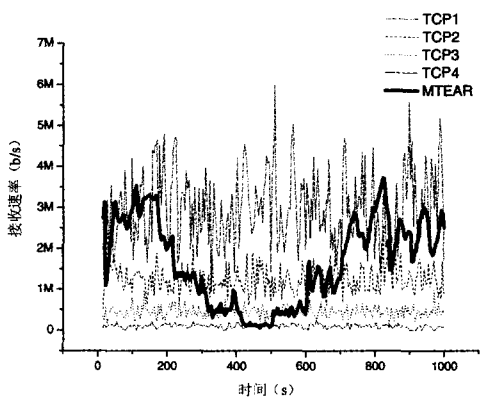

图 9　TEARM 和 TCP 的接收速率

当丢失率更高的接收端加入组播组，TEARM 可以很快选出合适的 CLR——链路丢失率最高的接收端，选择新的 CLR 的时延粗略地为 1～3s；发送速率也可以非常迅速地调整到与 TCP 流相当的水平(图 9)。发送速率随着丢失率高的接收端加入而不断下降，随着其退出不断升高。在 10s 内可以迅速调整到准确的速率。总之，TEARM 能够很好地响应不同程度的链路丢失率。

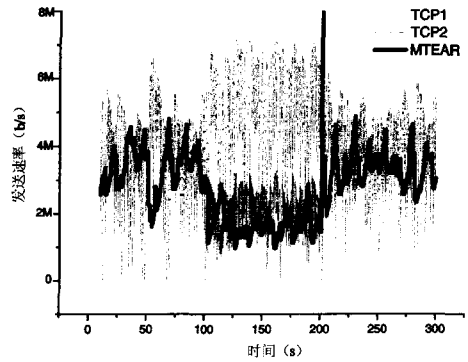

图 10　TEARM 和 TCP 的吞吐量

2.3.2　对低速带宽的响应性

采用哑铃型拓扑。TEARM 有两个接收端，接收端 R0 与一条 TCP 流竞争 8M 带宽。在 100s 时，接收端 R1 加入 TEARM 组播组，并且与一条 TCP 流竞争 4M 带宽，而该接收端在 200s 退出组播组。

由图 10 可以看出，低速的 R1 加入组播组后，TEARM 能够将其速率调整到合适的带宽，并且能够选出合适的代表作为当前受限接收端。TEARM 对速率的响应时间基本在 3s 以内，在 1s 内可以迅速选出 CLR。TEARM 可以对低速的接收端加入作出迅速的响应。但是当低速接收端在组播组内时，由于接收端 R0 长时间没有经历丢包，其 *cwnd* 值大于接收端 R0 实际的 *cwnd* 值。当接收端 R1 退出组播组时，因为接收端 R0 的估计速率高于其实际的带宽，导致发送端以过高的速率发送数据（图 10 中的尖峰）。但是根据接收端 R0 的反馈信息，发送速率很快就恢复到了公平带宽 4M。可以说，TEARM 对带宽具有良好的响应性。

总结和展望　TEARM 在接收端模拟了 TCP 的拥塞控制机制，而且对多个 epoch 的速率进行加权平均，估计吞吐量，并采用了代表机制来抑制反馈。通过仿真可看出，TEARM 与 TFMCC 相比，具有相同的 TCP 公平性和速率平滑性，并且在可扩展性方面优于 TFMCC，以及能够很好地响应不同的链路丢失率和带宽，使得 TEARM 适合于流媒体业务的传输。但是，由于 TEARM 是一个单速率的组播拥塞控制协议，在某些情况下，低速的 CLR 无疑会限制其他高速接收端的吞吐量，使得 TEARM 缺乏协议内的公平性；此外，对于低速接收端迟加入的情况，高速的接收端估计速率很可能要远大于实际的链路带宽，会对协议的性能造成潜在的威胁。因此，需要将 TEARM 扩展到多速率组播拥塞控制协议中，通过引入分层机制来进一步改善协议的性能。

参 考 文 献

1　石锋，吴建平. 组播拥塞控制综述[J]. 软件学报，2002，13(4)：1～9

2　Padhye V，Towsley Firoiu D，Kurose J. Modeling TCP Throughput：A Simple Model and its Empirical Validation. In：Proceedings of ACM Sigcomm，Vancouver，Canada，1998

3　Widmer J，Handley M. TCP-Friendly Multicast Congestion Control (TFMCC)：Protocol Specification [EB/OL]. draft-ietf-rmt-bb-tfmcc-01. txt. 2002

4　Widmer J，Denda R，Mauve M. A Survey on TCP-Friendly Congestion Control [J]. IEEE Network，2001，15(3)：28～37

5　Nonnenmacher J，Biersack E W. Scalable Feedback for Large Groups [J]. IEEE/ACM Trans on Networking，1999，7(3)：375～386

6　Bhattacharyya S，Towsley D，Kurose J. The Loss Path Multiplicity Problem in Multicast Congestion Control [A]. In：Proc. of IEEE INFOCOM [C]. New York：IEEE Communications Society，1999. 856～863

7　Rhee I，Ozdemir V，Yi Y. TEAR：TCP emulation at receivers-flow control for multimedia streaming：[NCSU Technical Report]. Apr. 2000

8　ns-2 Network Simulator. http://www. isi. edu/nsnam/ns，2002

9　Floyd S，Handley M，Padhye J，et al. TCP Friendly Rate Control (TFRC)：Protocol Specification. Request for Comments (RFC) 3448，The Internet Society，January 2003

（上接第 30 页）

可以看出，由于 MCOPS 算法是一种分布式路由算法，它不需要维持全局状态，仅仅依靠局部信息选择路由，因此在 Ad Hoc 网络高度动态的环境下，能够取得更好的路由成功率。

结束语　本文提出了一种基于多条件约束的最优路径选择算法 MCOPS（Multi-Constrained Optimal Path Selection），其基本思想是：初始计算从每个结点到给定的目的结点的最优路径代价，这条最优路径与各条边的权值及它们的线性拟合相关。然后，每条路径需要使用 $K+1$ 次 Reverse-Dijkstra 算法。随后，算法在图上随机地移动，发现那些到达最终目的结点的机会较大的结点。MCOPS 算法由于其利用局部状态信息分布式地进行路由选择的特征，使其具有可扩展性，同时，也适合于 Ad Hoc 网的高度动态的环境，并且容易扩展到其他 QoS 约束的路由问题。

由于 Ad Hoc 网络的动态性、状态信息的聚集性等因素，在同一时刻，对于每一个结点而言，网络的真实状态信息并非总是可用的。对于同一对源结点和目的结点，每次执行算法时，可能会产生不同的路径。因此，需要进一步研究路径选择策略对负载平衡的影响以及如何对网络状态信息的不精确性作出补偿，实现网络中的负载分担，减少了繁忙节点的负荷，从而延长整个网络的寿命，使得 MCOPS 在移动 Ad Hoc 网络环境下有更好的稳定性。

参 考 文 献

1　Kulpes R，Van Mleghem P，Korkmaz T，Krunz M. An Overview of Constraint-Based Path Selection Algorithms for Qos Routing. IEEE Communications Magazine，December 2002

2　Orda A. Routing with end-to-end QoS guarantees in broadband networks. IEEE/ACM Transmission Networking，2002，7(3)：365～374

3　Jaffe J M. Algorithms for Finding Paths with Multiple Constraints. Networks，1999，14：95～116

4　Iwata A，Chiang C C，Pei G，Gerla M，Chen T W. Scalable routing strategies for Ad Hoc wireless networks. IEEE Journal on Selected Areas in Communications，1999，17(8)：1369～1379

5　Sivakumar R，Sinha P，Bharghavan V. CEDAR：A core-extraction distributed Ad Hoc routing algorithm. IEEE Journal on Selected Areas in Communications，1999，17(8)：1454～1465

6　Jain R，Puri A，Sengupta R. Geographical routing using partial information for wireless Ad Hoc networks. IEEE Personal Communications，2001，8(1)：48～57

7　Paul K，Bandyopadhyay S，Mukherjee A，Saha D. A stability-based distributed routing mechanism to support unicast and multicast routing in Ad Hoc wireless network. Computer Communications，2003，24(18)：1828～1845