计算机科学2002Vol.29№.3

Internet 网络的 TCP 友好控制机制*)

TCP-Friendly Congestion Control in the Internet

王 茜[1] 隆克平[1] 程时端[1] 张润彤[2]

(北京邮电大学程控交换与通信网国家重点实验室 北京100876)[1](Nokia 中国研发中心 北京100013)[2]

Abstract This paper investigates several TCP-friendly congestion control mechanisms and sorts them according to their implementation methods. Advantages and disadvantages of every kind of these TCP-friendly schemes are listed after their principles. This paper also suggests that combination of end-to-end mechanisms and hop-by-hop mechanisms is the future of TCP-friendly congestion control as well as combination of AIMD-based scheme and model-based scheme.

Keywords TCP-Friendly, Congestion Control, Flow Control, QoS

1. 引言

随着 Internet 网络的飞速发展，传统的"尽力"型业务越来越不能满足新的用户和应用的质量要求，网络需要提供更高级的质量保障。同时，对于不同的用户对业务质量(QoS)具有不同的要求，网络也需要区别 QoS 等级，并采用相应的传输控制机制来保障这些 QoS 等级。在传统 Internet 网络中，分别用两种传输协议来传送不同质量要求的业务：TCP (Transmission Control Protrol)和 UDP(User Data Protocol)协议。其中，TCP 协议适用于突发量较大，需要确保发送而对时延要求不高的业务，如文件传输、数据传送等；UDP 协议适用于一些突发较小，对确保发送要求不高而对时延要求高的业务，如话音、视频等多媒体业务。由于两类业务具有不同要求，TCP 协议采用窗口机制来进行流量控制，通过每个连接的接收方反馈的信息来改变发送方的发送窗口大小，调整发送速率，以适应网络的负载状况，同时通过反馈信息确定数据包的正确发送，如果没有正确的反馈信息，则重传该包。UDP 协议则不具有流量控制和确保传送的相应机制，它不发送反馈信息，不会因为网络的拥塞而减少数据的发送速率，但它的优势在于没有 TCP 重传机制引起的网络额外负载，而且也不需要接收方发送确认信息。这些特性使 UDP 协议特别适用于日益增多的网络多媒体实时业务。

目前，具有拥塞控制的 TCP 连接仍然占有 Internet 网络的绝大多数业务量，但基于 UDP 的实时多媒体业务，如 IP 电话、组通信等都呈快速增长趋势，极有可能在一段时期后与 TCP 业务分庭抗争。通过大量的理论分析和网络实验，研究人员发现如果不考虑网络的实际容量而大量发送非拥塞控制的 UDP 业务，很容易导致网络过载和高丢失率，严重时会引起"拥塞崩溃"现象。另一方面，非拥塞控制业务还会大量侵占一丢包就减少发送速率的 TCP 业务的带宽，有时会导致 TCP 协议根本不能建立连接，更谈不上传送数据了。因此，基于 UDP 的业务需要增加相应的控制机制，不仅要避免网络过载，在一定程度上确保自己的 QoS 要求，还要与 TCP 业务公平地分享网络资源，这就是 Sally Floyd 等提出 TCP 友好(TCP-friendly)的概念[1,2]。这种 TCP 友好机制希望达到的目标是：

1)拥塞控制机制应保证网络不出现拥塞崩溃现象，使网络获得较高的利用率和较低的丢包率。

2)采用拥塞控制的应用比不采用拥塞控制的应用获得更高效的带宽利用，并获得更好的性能。

3)根据网络负载来调整速率的应用应该获得更多带宽的占用权，并在 Internet 中得到广泛运用。

4)网络应该要求所有的应用都开始进行拥塞控制，对那些不采用拥塞控制的应用被视为恶意应用，应受到严厉的惩罚，如网络拥塞时这些业务的数据包具有最高的丢失优先级。

针对这种 TCP 友好的概念，研究人员陆续提出了许多 TCP 友好控制机制。目前，国内外相关的研究主要是各种端到端的拥塞控制机制，包括基于 TCP 窗口机制的拥塞控制以及基于模型和公式的 TCP 友好控制机制等。另外还有一些逐跳(hop-by-hop)的控制机制，例如基于速率的逐跳拥塞控制机制等，我们对这些机制进行了分析和比较，提出了进一步研究的发展方向，希望能推进相关领域的研究。

2. 响应流与非响应流

为了简单描述采用拥塞控制的应用和不采用拥塞控制的应用，产生了一种定义：响应流与非响应流。响应流就是前面提到的类似 TCP 的具有端到端拥塞控制的应用流。相反，非响应流指的是不采用端到端拥塞控制的业务流，特别是丢包后也不会减少它们对网络的带宽占用的流。这些非响应流会给 Internet 网络带来拥塞崩溃和非公平性，非公平性是由于非响应流大量侵占响应流的带宽而造成的，拥塞崩溃则是由于网络大量传送的非响应流数据包会在下游的拥塞节点丢失而造成的资源极低效运用。另一种更严格的区分定义是：TCP 友好流和非 TCP 友好流。TCP 友好流的定义是长期的到达速率不大于任何在同等网络条件下的 TCP 流的速率的业务流，非 TCP 友好流则正好相反。从以上定义可以看出，非响应流一定是非 TCP 友好流，而非 TCP 友好流不一定是非响应流。以下我们主要讨论的是对定义更广的非响应流的 TCP 友好拥塞控制。

目前广泛应用的 Reno 版本的 TCP 协议是通过 AIMD (Additive-Increase Multiplicative-Decrease)机制来实现对网络拥塞的反应和控制。也就是说，当接收方没有反馈丢包信息

*)本研究得到诺基亚研究中心合作项目和国家自然科学基金(项目批准号69972008)联合资助。

时，TCP 的发送窗口会按线性过程 I 增加，而一旦接收方信息指示网络有丢包时，发送窗口会减半，发送速率呈指数递减 D 的状态。这个过程可以用 AIMD(α,β)及式(1)表示：

$$I:\omega_{t+R}\leftarrow\omega_t+\alpha;\alpha>0$$
$$D:\omega_{t+\delta t}\leftarrow(1-\beta)\omega_t;0<\beta<1 \quad (1)$$

其中，$\alpha=1$个包，$\beta=1/2$，ω_t 是窗口在时刻 t 的大小，R 是 RTT (round trip time)时间。

虽然 TCP 的 AIMD 拥塞控制机制能很好地控制块数据的传送，但如果多媒体业务检测到拥塞指示时也将发送速率减半的话，则会极大地降低用户对实时业务预期的质量。因此，我们不能直接将 TCP 的拥塞控制机制应用到多媒体等实时业务上，而应该针对它们对时延敏感的特性设计适合的控制机制。也就是说，为非响应流设计 TCP 友好拥塞控制机制。目前，国际国内研究人员通过深入研究 TCP 业务的流量特性，提出了以下多种 TCP 友好控制机制。

3. 端到端的 TCP 友好控制机制

3.1 基于 AIMD 的控制机制

这类控制机制还可以根据其实现手段分为基于窗口的和基于速率的控制机制两大类。

3.1.1 *基于窗口的 AIMD 控制* 在上节中已经提到，不能直接将 TCP 的 AIMD 控制机制用于实时业务的控制。文[3]将 TCP 的 AIMD 控制表达式(1)改进为：

$$I:\omega_{t+R}\leftarrow\omega_t+\alpha/\omega_t^k;\alpha>0$$
$$D:\omega_{t+\delta t}\leftarrow\omega_t-\beta\omega_t^l;0<\beta<1 \quad (2)$$

并推导出只要 $k+l=1$且 $l\leqslant1$，这样的控制算法就是 TCP 友好的控制机制。从式(2)中可以看出，k 越大，速率增加的幅度越小，则业务流对带宽的侵略性越小。l 越大，速率减小的幅度越大，就越不能满足业务的实时性要求。对于 TCP 本身的控制机制而言，$k=0$，$l=1$。对于实时流，文[3]认为 $l<1$的控制机制就可以避免丢包时发送速率减半的情况而提供好的业务性能。同时，作者提出了两个实现算法用于实时流的控制，即 IIAD($k=1;l=0$)和 SQRT($k=1/2;l=1/2$)。仿真结果证明这两种算法可以与 TCP 的 AIMD 算法公平地共享带宽资源，同时还能为各自的应用提供所需的性能保障。

文[4]中推导了 AIMD(α,β)控制下的业务流吞吐量与 α，β，RTT 以及丢包率 p 的关系函数近似为：

$$T_{\alpha,\beta,R,p}=\frac{\sqrt{2-\beta}\sqrt{\alpha}}{\sqrt{2\beta}RTT\sqrt{p}} \quad (3)$$

文中还计算了 TCP($\alpha=1;\beta=1/2$)时的吞吐量公式，并与式(3)结合，得到 TCP 友好业务流应满足的关系式：

$$\alpha=3\beta/(2-\beta) \quad (4)$$

这样，只要根据 TCP 的控制原理和式(4)，就可以得到一系列 TCP 友好的控制机制。

基于窗口的 AIMD 控制机制的优势在于它的工作原理以及稳定性、公平性等特性已经被大家所熟识，实现非常简单。但它的缺陷在于发送速率改变的幅度过大，业务流的突发性强，抖动大。只适用于希望以 TCP 友好方式尽快发送数据，而对速率变化程度不做太大要求的业务，也就是那些对实时性要求不是很高的业务。这类算法的另一个问题在于，如何针对不同性能要求的业务设计合适的参数还没有通过理论分析或实验得到理想的依据。

3.1.2 *基于速率的 AIMD 控制* 其代表算法有 RAP (Rate Adaptation Protocol)和 LDA (Loss-Delay based Adjustment algorithm)。

RAP[5]是针对拥塞控制和丢失检测来设计的，并且采用分层编码来调整业务质量。RAP 的实现主要是利用源端的包序号和终端的 ACK 应答来建立端到端的反馈。RAP 与窗口机制不同的是它不是由 ACK 的到达来触发包的发送，而是利用 ACK 反馈来检测包丢失并抽样出 RTT 值，并通过这些值来计算下一个包的发送间隔，利用计时器来触发包的发送。

RAP 采用与 TCP Reno 相同的丢包检测机制，即连续收到三个相同的 ACK 反馈则证明有丢包，同时也支持反馈超时。当检测到丢包时，发送速率按式(5)指数减少：

$$S_{t+1}=\beta S_t,IPG_{t+1}=IPG_t/\beta,\beta=1/2 \quad (5)$$

当网络无丢包时，发送速率按下式线性增加：

$$S_{t+1}=\frac{PacketSize}{IPG_t};\ IPG_{t+1}=\frac{IPG_t*C}{IPG_t+C} \quad (6)$$

其中 S_t 是当前的发送速率，S_{t+1}是下一阶段的发送速率，IPG 是两个包之间的发送间隔。C 是一个常数，用于调整发送速率的线性增加量。

LDA[6]则利用了目前用于实时流传送的 RTP(Real Time Protocol)和 RTCP(RTP Control Protocol)协议来传送网络状况参数。RTP 和 RTCP 是建立在 UDP 等非响应传输协议上层，用于连续媒体流的控制和传送的协议。业务的源端和终端通过 RTCP 的控制消息来传送报告。源端的报告中包括发送数据的数量和报告生成时的时间标签，可能还带有检测包的序号，终端收到这个报告后就返回一个描述丢失包的分额，上一次收到报告的时间以及自己处理报告所需时间的反馈报告。源端利用这个反馈报告就可以计算出丢包率 l，RTT 时间并估计检测包在传输过程中的瓶颈带宽 b。

LDA 源端为发送速率设置一个线性增加值 AIR(Additive Increase Rate)，并设很小的初值。利用 RTCP 报告获得的估计信息，当网络无丢包时，AIR 和发送速率 r 按式(7)变化：

$$AIR_{t+1}=AIR_t\times(1-\frac{r_t}{b})\quad r_{t+1}=r_t+AIR_t \quad (7)$$

如果网络有丢包时：

$$r_{t+1}=r_t\times(1-(l\times R_f)) \quad (8)$$

其中，R_f 是速率减小因子，决定了源端对丢包的反应程度。

这类基于速率的 AIMD 算法同样也能获得不同流间的公平的带宽共享，以及较低的丢包率和较高的带宽利用率。它们的优势在于是对发送速率的调整，而不会象基于窗口的机制那样造成大的抖动，同时也不是基于 ACK 触发的发送，不会产生突发大的业务流。它们的问题在于每个丢包都引起发送速率的减少，对于实时流的控制也不是很合适，也会造成一定程度的质量降低。因此，如何平滑地调整非响应流的发送速率，使之成为 TCP 友好的业务流，就成为基于建模的控制机制的研究出发点。

3.2 基于建模的控制机制

基于建模的算法是在对 TCP 流量进行建模后提出的一系列控制机制。文[7]中推导出 AIMD(1,1/2)控制下的 TCP 流量满足

$$T=\frac{B}{R\sqrt{2/3p}+t_{RTO}(3\sqrt{3p/8})p(1+32p^2)} \quad (9)$$

其中，B 是平均包长，R 和 p 是稳定状态下的 RTT 和丢包率，t_{RTO}是 TCP 的重传超时时间。以下的控制机制都是基于这个模型而得到的，所以可以统称为基于建模的控制机制。

3.2.1 *基于公式的控制机制 TFRC* S. Floyd 根据式

(9)提出了一种基于公式的控制机制 TFRC(TCP 友好速率控制机制)[8]。这种机制的目的是要保持一个相对稳定的发送速率,同时对拥塞做出反映。

支持 TFRC 的业务源端和终端分别执行不同的功能:终端测量包在两端间传送所需的 RTT 时间,同时计算反映连续的多个 RTT 的丢失事件的速率,这种丢失事件指一个 RTT 时段内的一个或多个连续的丢包。终端在每个 RTT 内要至少一次向源端反馈收包的信息。利用获得的以上参数,终端按式(9)计算出源端允许的发送速率 T,并将这个值发送给源端。源端收到信息后就会按照 T 直接增加或减少发送速率,而不是增加或减少窗口大小了。

TFRC 一方面不象非响应流那样侵略性地抢占可获得的带宽,而是根据丢失事件速率的减小而平滑地增加发送速率;另一方面,它也不会因为单个包的丢失而将速率减半造成抖动,只是在多个连续的丢失事件发生后才减小一半的速率,比较适合实时媒体流的流量控制。

3.2.2 *基于模型的控制机制* TFRCP TFRCP 也是基于文[7]中 TCP 建模的另一表达式:

$$T \approx f(W_{max}, R, p, t_{RTO}) \quad (10)$$

其中,R,p,t_{RTO}与前一节中相同,而 W_{max}是终端接收窗口的最大值。

TFRCP 源端按固定时间周期计算发送速率并依此计算该周期内可以均匀发送的包数,用计时器触发包的发送。每个源端发出的包携带着序号和发送的时间标签,终端对每个包进行 ACK 确认,确认中包括确认的序号和确认的时间标签。式(10)中 R 和 t_{RTO}由这些时间标签来计算,p 的计算基于一个周期内丢失的包数和已得到确认的包数。如果丢失的包数为 0,则 $T_{i+1}=2\times T_i$;如果丢失的包数大于 0,则 $T_{i+1}=f(W_{max}, R, p, t_{RTO})$。由于 TFRCP 采用了固定时间周期的计算方法,使各种相关参数的计算相对简单了许多,实现起来也比较容易。但它的问题在于无丢包时,速率呈指数增长,业务量的突发较大,对剩余带宽的侵占性也较大。

总的来说,基于建模的控制机制平滑了丢包情况下时的速率降低,避免了 AIMD 机制的速率抖动大,实时业务性能不佳的情况。但问题在于 TCP 建模的准确性以及各种参数的测量都会影响控制机制的 TCP 友好性能。文[7]中的 TCP 模型是在单连接、稳定丢包的状态下推导出来的,不能正确地反应各种状态下 TCP 的流量特性,特别是在丢包率较大的时候,利用这些模型的控制机制不再呈现 TCP 友好的特性了。因此,还有一类控制机制将建模与 AIMD 机制结合起来,用于解决这种问题。

3.3 建模与 AIMD 机制的结合

将建模与 AIMD 机制的结合的控制机制的代表是 DAA (Direct Adjustment Algorithm)[10]。DAA 也是利用 RTP 和 RTCP 协议来进行状态检测和流量控制的,同时还针对组播的情况进行考虑。文[10]中指出,如直接用 AIMD 调整速率,由于 RTCP 的控制信息发送不频繁,不能及时反应网络丢包的情况,而使控制机制不能达到 TCP 友好的性能。但如果用 TCP 建模的近似公式:

$$T=1.22\times B/R\times\sqrt{p} \quad (11)$$

当丢包率 $p>16\%$时,这个公式就不能正确反应 TCP 的流量特性了。于是文[10]将这两种机制结合起来,取长补短。

DAA 中,RTP 终端发送的 RTCP 控制报告包含两次控制报告间的丢包率和时间标签,RTP 源端收到 RTCP 报告后根据丢包率来确定该增加还是减少发送速率。具体步骤如下:1)计算源端与终端间的 RTT;2)采用低通滤波器算法计算平滑的丢包率 p;3)当 $p\leqslant 16\%$时,按式(11)调整源端发送速率;当 $p>16\%$时,源端按 MD(1/2)来减小发送速率,即速率减小一半。

这种机制结合了建模与 AIMD 算法的优点,相对地避免了这两种机制独立存在时的缺点,能提供更有效的 TCP 友好控制效果。但在丢包率较大时,仍然造成速率的指数减小,使实时业务的性能降低。我们认为当丢包率 p 较大时,可以按 3.1节中提到的几种方式来平滑地减小速率,提供更好地实时媒体流性能。

4. 逐跳的 TCP 友好控制机制

端到端的控制机制虽然可以对网络拥塞做出反应而控制业务流量,但由于端到端的反馈信息只能定期地发送而不能及时地反应拥塞。相比之下,逐跳的控制机制可以作用在汇聚业务流的拥塞节点,每一跳都可以按照实际的流量负荷来反馈信息,它的反馈周期更短,控制更快,更精确。

4.1 基于速率的逐跳控制机制

在基于速率的逐跳控制机制[11]中,每个包交换机都控制单个流的输出速率。交换机根据每个流在输出队列中的缓冲区占用量来估计每个流的流量,并周期地向最相邻的上游交换机传送这些流量特性。上游交换机的主要作用是调整输出速率,以适应下游的容量。这里,业务源端也被看作是一个上游交换机来调节发送速率。上游交换机收到一次下游的反馈信息就重新计算一次最大的发送速率:如果观察到的前 k 时刻的缓冲区占用量小于1,则速率呈线性增长;否则按估计的缓冲区占用量和期望的占用量的公平共享值以及之前各时刻速率的加权平均值来计算新的速率。

文[11]中提议的逐跳速率控制算法考虑的是各种业务流在网络中间节点的拥塞控制,可以适用于非响应流。但它还没有与 TCP 友好的要求结合起来,提供的业务流还不能算是 TCP 友好的业务流。在这方面还可以做进一步的研究。

4.2 模拟 ATM 业务流的控制机制

ATM 传输技术的优势在于它的拥塞控制和流量控制可以很好地提供业务质量的保障。因此,许多 IP 网络中的控制机制都从 ATM 技术中获得启发而产生。ALS(Adaptive Load Service)[12]就是模仿 ATM 中的 ABR 业务流的一个例子。

ALS 也是通过 RTP 和 RTCP 消息来建立控制机制的。ALS 源端周期地发送控制信息,指示它所需要的传输速率。网络的中间路由器根据自己可获得的资源修改这个参数,然后将控制信息逐跳地发送到终端。终端同样也可以根据自己的情况调整这个参数,并将其返回给源端。源端则根据返回值调整其发送速率。ALS 也很适合组播多媒体业务流的控制和质量保障。

从以上两类控制机制我们可以看到,逐跳的拥塞控制机制可以较准确地跟踪网络内部的拥塞情况并及时做出控制反应。同样可以使网络获得较高的利用率和较低的丢包率,也使采用拥塞控制的应用与不采用拥塞控制的应用获得公平的带宽利用。但它们只保障了逐段的业务性能,却不能保障业务流端到端的性能,也不能从端到端的角度提供 TCP 友好的业务流量特性,不能算是完全实现了 TCP 友好控制。

结论与未来的研究方向 随着网络技术的不断发展,实

(下转第77页)

对接收到的数据的进一步转发处理过程如下：

·如果本会议小组内还有其它 BAMClient 使用了同一 BAMCU，则 BAMCU 向这些节点转发数据。

·如果本会议小组内有其它 BAMCU 存在，则为本 BAMCU 分配向其它 BAMCU 转发数据所需要的转发端口并开始转发数据。其它，BAMCU 对接收到的数据根据相应的 BAMClient 的性质不同分别启动本地组播和远程定向发送数据进程。

b）数据接收　如果同一多媒体会议小组，除去自己外还有别的节点使用该 BAMCU 来转发多媒体数据，则所需做的只是把 BAMCU 接收到的其它 BAMClient 节点的多媒体数据转发过来即可。

如果同一多媒体会议小组中，只有新加入节点自己采用该 BAMCU 来转发多媒体数据，那么还必须把其它所有节点的多媒体数据首先转发到新加入节点对应的 BAMCU，然后再转发到新加入节点 BAMClient。

4. 应用例子

采用本文提出的基于 Agent 的 CSCW 多媒体交互环境模型，我们实现了一个支持多媒体交互的协作设计系统—CoopDesigner。CoopDesigner 系统提供了一个包括多媒体交互环境、电子白板、混合机制应用共享等协作工具的基本协作环境，用户可以在这个环境中根据自身需求灵活裁减和配置系统组成。CoopDesigner 系统向用户提供了基于 IP 网络的全双工视音频交互功能，并且通过 Agent 间的协商交互，系统可以动态确定多媒体数据的转发路径，实现垮物理网段的多媒体数据的多点传送。

结论　多媒体交互是 CSCW 中的一种重要的交互协作方式。但现有 CSCW 的多媒体交互环境中对多媒体数据的跨网段传输采用的基本都是静态的 MCU 分布结构，使得当局部 MCU 节点出现故障后，不能及时建立新的多媒体数据的传送通道。而我们提出的基于 Agent 的多媒体交互模型中，多媒体数据转发路径是通过 Agent 间的协商交互动态建立的，当局部多点控制单元出现问题后，系统可以迅速建立新的多媒体转发路径，因而基于该模型的系统具有较强的的可靠性和适应性。

参 考 文 献

1 Grudin J. Computer-supported cooperative work: History and focus. IEEE Computer, 1994, (5): 19～26

2 郑庆华. CSCW 的理论、技术与实现：[博士论文]. 西安：西安交通大学，1997

3 倪强，朱光喜. 计算机支持下的协同工作的研究现状综述. 计算机工程与应用，2000，36(4)

4 Finin T, Fritzson R, et al. KQML as an agent communication language. In: the Proc. of the third intl. conf. on information and knowledge management, ACM press, Nov. 1994

（上接第56页）

时多媒体业务成为网络进一步的研究重点。当前存在的问题主要是：如果不考虑网络的实际容量而大量发送非拥塞控制的 UDP 业务，很容易导致网络过载和高丢失率，严重时会引起“拥塞崩溃”现象；非拥塞控制业务还会大量侵占—丢包就减少发送速率的 TCP 业务的带宽，有时会导致 TCP 协议根本不能建立连接，更谈不上传送数据了。因此，Sally Floyd 等提出 TCP 友好（TCP-friendly）的概念，并由此产生了一系列 TCP 友好控制机制。本文从实现方式的角度，将这些 TCP 友好控制机制总结成几类，并分析了它们的优势和不足。通过文中分析，我们可以看到基于 AIMD 和基于建模的控制机制各有利弊，但如果将两者结合，可以更好地把握流量的控制力度，既实现了 TCP 友好的流量控制，又满足了实时媒体流的性能要求。从另一方面来说，端到端的机制与逐跳的机制也是这样。端到端的机制对业务端到端的流量进行控制，提供 TCP 友好的流量特性，并保障实时业务的质量，但它对网络拥塞的反应较慢，控制不及时；逐跳的控制机制对网络内部的拥塞状况了如指掌，采取控制时快速，准确，但对业务流的端到端性能不能很好地把握，也不能完全实现 TCP 友好的控制。因此，最佳的方案是将这几种方式结合起来，提出全局 TCP 友好的控制机制，这样，不仅可以更好地实现网络拥塞控制，对于实时媒体流而言，还能提供 TCP 友好的流量特性，确保网络响应流与非响应流间的公平性，保障网络的总体性能。

参 考 文 献

1 Floyd S, Fall K. Promoting the use of end-to-end congestion control in the Internet. IEEE/ACM Trans. On Networking, 1999, 7 (4)

2 Floyd S. Congestion control principle. IETF draft (draft-floyd-cong-02.txt), March 2000

3 Bansal D, Balakrishnan H. TCP-friendly congestion control real-time streaming application: [MIT technical report]. May 2000

4 Floyd S, Handley M, Padhye J. A comparison of equation-based and AIMD congestion control. May 2000. URL http://www.aciri.org/tfrc

5 Rejaie R, Handley M, Estrin D. RAP: an end-to-end rate-based congestion control mechanism for realtime streams in the Internet. In: Proc. of INFOCOMM'99, volume 3, March 1999

6 Sisalem D, Schulzrinne H. The loss-delay based adjustment algorithm: a TCP-friendly adaptation scheme. In: Proc. of NOSSDAV'98, Cambridge, England, July 1998

7 Padhye J, et al. Modeling TCP throughput: a simple model and its empirical validation. In: Proc. of SIGCOMM'98, Aug. 1998

8 Floyd S, Handley M, Padhye J. Equation-based congestion control for unicast application. In: Proc. of ACM SIGCOMM 2000, Aug. 2000

9 Padhye J, et al. A model based TCP-friendly rate control protocol. In: Proc. of NOSSDAV'99, June 1999

10 Sisalem D, Emanuel F, Schulzrinne H. The direct adjustment algorithm: a TCP-friendly adaptation scheme. August 1997. URL: http://www.focus.gmd.de/usr/sislem

11 Mishra P, Kanakia H. A hop by hop rate-based congestion control scheme. In: Proc. on Communications Architecture & Protocols. Baltimore, MD USA. Aug. 1992

12 Sisalem D, Schulzrinne H. ALS: an ABR-like service for the Internet. In: Proc. of ISCC'2000, France, July 2000