数据包时延及控制策略的研究

胡金初
(上海师范大学数理信息学院　上海 200234)

摘　要　许多网络应用使用 TCP 协议，为了能够获得可靠的数据传送服务，作为开发人员除了关心可靠性外，还要考虑拥塞控制的问题，本文中提到的拥塞控制窗口，能够实现数据速率的控制。TCP 协议采用慢启动和拥塞避免策略实现端到端的数据传送。

关键词　延迟，窗口策略，慢启动，拥塞控制

Study of Packet Delay and Control Strategy

HU Jin-Chu
(Shanghai Normal University, Shanghai 200234)

Abstract　Many network applications run over TCP rather than UDP because they want to make use of TCP's reliable transport service. But an application developer choosing TCP gets not only reliable data transfer but also TCP congestion control. We have just seen how TCP congestion control regulates an application's transmission rate via the congestion-window mechanism. TCP implements an end-to-end congestion-control mechanism with slow start and congestion avoidance.

Keywords　Delay, Windows strategy, Slow start, Congestion-control

1　引言

随着 Internet 的发展，它的拥挤状况也日益严重，当通信子网中数据包的数量太多造成网络通信不畅时，网络性能就会降低，类似于交通阻塞一样。造成拥塞有许多因素，比如没有足够的空间来保存同时要求传输的数据包，有些数据包就会丢失。在某种程度上增加内存会有所帮助，但是如果路由器的存储空间增大到一定程度，队列的长度就会增加，可能会引起大量数据包的超时而造成重发数据包，反而会使拥塞变得更严重。又如路由器的 CPU 处理速度太慢，以至于不能完成正常的工作，那么即使有多余的通信线路也可能造成拥塞。类似的还有通信线路的带宽不够也会造成拥塞。这些因素都会造成数据包的丢失，问题严重时更会造成恶性循环，最终导致网络的崩溃。拥塞控制和流量控制存在着不同。拥塞控制的功能是必须保证通信子网能运送要传输的数据，这是有关全局性的问题，涉及到系统中的所有主机、路由器和相关设备等，考虑的是一个综合性因素。而流量控制只限于某个发送方和接受方之间点到点的通信量控制。它要保证一个快速发送者不能以比接收者能承受的速率更高的速度传输数据，简单地说就是要解决网络中出现快发慢收的问题。

按照控制论的观点，将拥塞控制算法分为两类：一类是开环控制方式，即不接受反馈的控制方式；另一类是闭环控制方式，根据网络的反馈不断地调整控制的方法和力度。开环控制方式就是尽量通过良好的设计来避免拥塞的发生。这些良好的设计包括何时接收新的数据包，何时丢弃何种类型的数据包等，通过这些设计使得拥塞问题在网络中不易发生。

2　数据包时迟分析

计算机网络系统是由许多硬件和软件组成的，是一个很复杂的信息系统，它要用高度结构化的方式来进行设计。现代的网络四通八达，从源计算机发出的数据包往往要经过几个路由器才能到达目的计算机，数据包在经历从一个路由器传到另一个路由器的过程中，由于数据包在处理的每一个环节都存在延迟，造成了数据包在两个路由器之间传输时延。它的延迟时间由四部分组成：节点处理时间 t_1、排队等待时间 t_2、传输时间 t_3 和传播时间 t_4。如果一个数据包从源传输到目的地在网络中经过了 k 个路由器，造成的总时间延迟 t 为：

$$t=\sum_{j=1}^{k}\sum_{i=1}^{4}t_{ij}$$

假设数据包的长度为 F、数据链路的速率为 R、数据包队列的平均长度为 v、信号的传输速率为 S、通信线路的长度为 D。

节点处理时间 t_1：路由器接受到一个数据包以后，检查和分析数据包的头，决定从哪一条输出链路上输出该数据包，并且作相应处理需要的处理时间。它由路由器的处理能力决定，一般路由器可以在微秒级时间内处理完。

排队等待时间 t_2：如果输出链路上有多个数据包在等待输出，则需要排队等待，根据队列的长短，需要等待一段时间，一般可以由 $t_2=vF/R$ 公式计算。t_2 会有很大的变化，如果队列是空，等待时间为 0，反之流量较大，队列也就比较长，则可能要等待较长的时间。

传输时间 t_3：网络中一般采用先来先服务的方式，路由器中的数据包经过排队，到达输出端口时，开始一位一位地传输数据，直到将整个数据包发送完为止，$t_3=F/R$。

传播时间 t_4：一旦数据位进入链路，就以电波的速度向前推进，由于两个路由器相隔一段距离，需要一定的传播时间，$t_4=D/S$，其中 S 在不同的介质中的速率是不相同的，一般 S 在 $2\times10^8\sim3\times10^8$ 米/秒。

3 AIMD 控制策略

传输控制协议 TCP 的目的是在不可靠的网络层上提供可靠的端到端通信。通过在发送方和接收方分别创建套接字的通信端点来获得 TCP 服务。每个套接字有一个套接字序号，它包含了主机的 IP 地址以及一个主机本地的 16 位号码。为了获得 TCP 服务，必须在发送方的套接字与接收方的套接字之间建立一个连接。一个套接字有可能被多个连接同时使用，这时用两端的套接字标识符来识别。

TCP/IP 协议族是因特网的核心。在 Internet 上大量的数据流使用的是 TCP/IP 协议，互连协议 TCP/IP 的拥塞控制机制对网络具有重要的意义。拥塞控制是确保 Internet 能够稳定和通信流畅的关键因素，也是其他管理控制机制和应用的基础，如多媒体通信中的 QoS 控制。是当前网络研究的一个热点问题。在 TCP 中，慢启动是发送者在刚开始数据发送时，为试探网络的拥塞状况而采用的试探性窗口增长策略。发送者先将控制窗口大小设为 1，如果没有发生数据包丢失，则每个发送周期都将窗口的大小增加 1 倍，发送周期是指从开始发送这个窗口的报文到收到该窗口内第一个数据的应答时间。当发送方发现数据包已经丢失，则将当前的窗口大小减半，结束慢启动的过程，采用 AIMD 的窗口调控策略。TCP 是通过动态控制滑动窗口的大小来解决拥塞现象的。它的拥塞控制算法需要有三个参数：接收方窗口、拥塞窗口和临界值。临界值在初始化时设置为 64kB。首先拥塞窗口保持指数规律增大，直到数据传输超时或达到接收方设定的窗口大小。也就是说，如果发送的数据长度序列，如 1024，2048 都能正常工作，但发送 4096 字节的数据时出现了定时器超时。那么就将临界值设置为当前拥塞窗口大小的 1/2，将拥塞窗口恢复成为最大数据段的大小，即 2048 字节。窗口的大小是按指数规律增长的，直到临界值这点为止，以后的数据传输要求拥塞窗口的大小再按线性规则逐渐增加。网络产生拥塞的根本原因在于用户的负载大于网络资源的容量和处理能力，结果造成数据包延时增加、丢包概率增大、应用系统的性能下降等。拥塞的原因有：

(1)带宽容量不足：高速数据流输入到低速链路上会产生拥塞，所有信源发送的总速率 ΣR 必须小于或等于信道容量 C。如果 $\Sigma R>C$，在理论上无差错传输则是不可能的。在网络低速链路处会形成带宽瓶颈，当满足不了所有源端带宽的要求时，网络就会发生拥塞。

(2)处理器处理能力弱、速度慢：如果路由器的 CPU 在执行排队缓存、更新路由表等处理时，其速度跟不上高速链路的要求，也会产生拥塞。

(3)存储空间不足：如果几个输入数据流共同需要同一个输出端口，那么在这个端口就会建立队列。如果没有足够的存储空间，排在后面的数据包则会被丢弃，尤其是突发的数据流。在一定范围内适当增加存储空间可以缓解这矛盾。但是存储空间也不是越大越好，路由器的存储空间超过某个范围时，拥塞只会变得更加严重。因为在路由器里数据包经过长时间排队完成处理时，它们早已超时了，而源端计算机误认为它们已经被丢弃而造成新的重发，实际上这些超时的数据包还会继续往下一个路由器转发，浪费网络资源，加重网络拥塞。

要避免拥塞的发生，需对以上原因进行综合考虑。低速链路配置高速 CPU 也会产生拥塞，提高链路速率而不改变处理器，会转移网络瓶颈，而不能避免拥塞。从另一个角度看，拥塞也是网络系统各部分不匹配的结果。拥塞一旦发生，往往会不断加重，形成一个恶性循环。如果路由器没有空余的缓存，那么它就必须丢弃新到的数据包。当数据包被丢弃时，源端会因超时而重传该包。由于没有得到确认，源端只能保留数据包，结果缓存会进一步消耗，并加重拥塞。

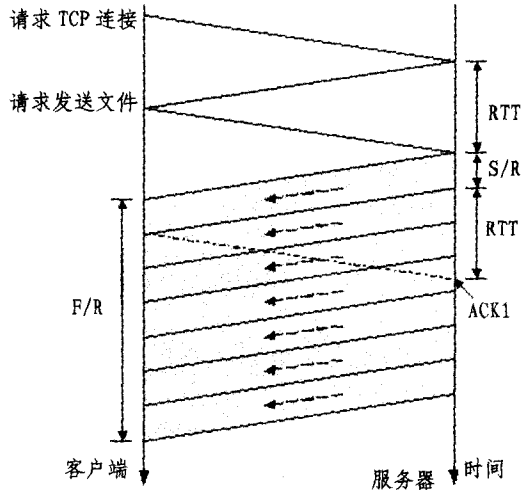

图 1 当 $WS/R>RTT+S/R$ 数据包传输分析

目前在 Internet 上使用的拥塞控制是建立在 TCP 的窗口控制基础之上的，IP 层的路由器所起的作用比较小，最近 IP 层控制拥塞的研究有逐渐增多的趋势。也就是说，该算法当出现了定时器超时，就将拥塞窗口减小 1/2 以后继续看是否满足传输要求。如果能够满足的话，再从此处开始逐渐增大滑动窗口的大小，如果一直不出现超时现象，拥塞窗口会增大到和接收方窗口的大小一样。此后，拥塞窗口将停止增大，只要不出现超时，并且接收方窗口也保持不变，那么拥塞窗口也保持不变。

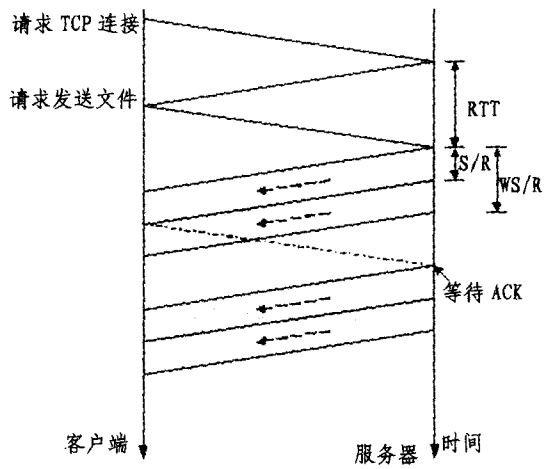

图 2 当 $WS/R<RTT+S/R$ 数据包传输分析

假设最大的数据段长度为 1024 字节。初始时拥塞窗口的临界值设置为 64kB，如果此时出现了超时，就将临界值设置为 32kB。拥塞窗口从 1kB 重新开始，之后如果没有出现超时，拥塞窗口的大小就按指数规律增大，一直到达临界值。此后就再按线性规律增加。当传输到一定时候出现了超时，临界值就被设置为当前的 1/2，例如是 20kB(因为出现超时的窗口值为 40kB)，并且又从头开始慢速启动的过程。客户端发

(下转第 101 页)

2 Li H F, Fan Y S. Workflow Model Analysis Based on Time Constraint Petri Nets [J]. Journal of Software, 2004, 15(1):17～26
3 Eder J, et al. Temporal Modelling of Workflows with Conditional Execution Paths [J]. In: M. Ibrahim, J. Kung, and N. Revell, eds. DEXA2000, LNCS1873, Springer-Verlag, 2000. 243～253
4 Combi C, Pozzi G. Temporal Conceptual Modelling of Workflows [J]. In: I.-Y. Song et al, eds. ER2003, LNCS2813, Springer-Verlag, 2003. 59～76
5 Eder J, et al. Time Constraint in Workflow Systems [J]. In: M. Jarke, A. Oberweis eds. CaiSE'99, LNCS1626, Springer-Verlag, 1999. 286～300
6 Bettini C, et al. Temporal Reasoning in Workflow Systems [J]. Distributed and Parallel Databases, 2002, 11:269～306
7 董威, 等. UML-Statecharts 的模型检验方法 [J]. 软件学报, 2003, 14(4):750～756
8 Alur R. Timed Automata [C]. In: 11th International Conference on Computer-Aided Verification, LNCS1633, Springer-Verlag, 1999. 8～22
9 Alur R, et al. Model-checking in dense real-time [J]. Information and Computation, 1995, 104:2～34
10 Berard B, et al. Systems and Software Verification-Model checking Techniques and Tools [M]. Springer-Verlag, 2001. 67～68

(上接第 53 页)

起 TCP 连接以后，请求从服务器传输一个对象，如果 $WS/R > RTT+S/R$，服务器就可以连续发送完整个对象(见图 1)，不需要中途停顿。反之，如果 $WS/R < RTT+S/R$，服务器为了预防网络拥塞的发生就必须中途停止发送(见图 2)，等待对方的应答，然后才能继续。下面分析一个数据包在 Internet 上传输时的控制情况。

设：网络来回的传输时间为 RTT (Round-Trip Time)；

传输文件的大小为 F 位；

服务器到客户端的数据链路的传输速率为 R bps；

拥塞控制窗口的大小为 W；

最大的数据段由 S 位组成；

则：发送一个数据段的时间 $=S/R$，如果 $WS/R>RTT+S/R$，可以将一个文件全部发送完，网络的反应时间 T 为：$T=2RTT+F/R$；

如果 $WS/R<RTT+S/R$，则服务器在传输中途必须停止，等待客户端的应答后才能进一步传输，这时的反应时间 T 还需要加上服务器等待的时间，为：$T=2RTT+F/R+(K-1)(S/R+RTT-WS/R)$；其中 $K=\lceil F/WS \rceil$，$K-1$ 可以看作服务器停顿的次数。考虑网络中可能会出现这两种情况，括号中的数值只取正值，得到：

$$T=2RTT+F/R+(K-1)(S/R+RTT-WS/R)^{+}$$

以图 3 为例，服务器从开始发送到收到应答信号的时间 $=S/R+RTT$；$F/S=15$ 段；$K=4$；第 k 个窗口的发送时间 $=2^{k-1}*S/R$；$i=\min\{K-1,Q\}$，其中 i 是服务器端的 TCP 连接停顿的次数，这里 $i=2$。

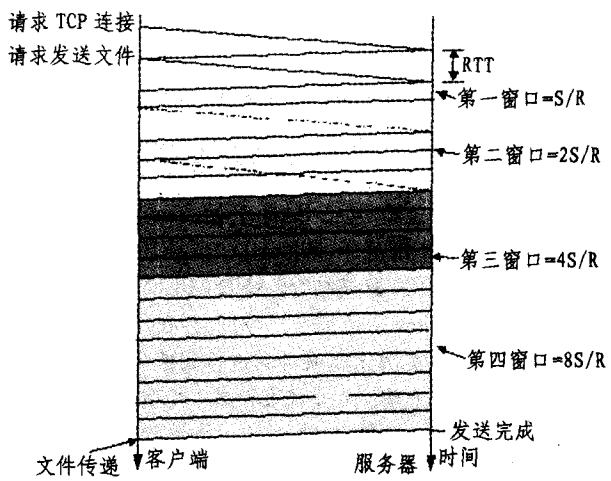

图 3 TCP 慢启动控制过程

$$T=\frac{F}{R}+2RTT+\sum_{j=1}^{i}T_{STOP_j}$$

$$=\frac{F}{R}+2RTT+\sum_{j=1}^{i}\left[\frac{S}{R}+RTT-2^{j-1}\frac{S}{R}\right]$$

$$=\frac{F}{R}+2RTT+i\left[RTT+\frac{S}{R}\right]-(2^{i}-1)\frac{S}{R}$$

目前 IP 在处理拥塞方面采用的方法有：

· Internet 使用最广的方式是先进先出法，其优点是处理简单。其实质是一种“去尾”的算法，所以有时在包丢失的公平性方面存在问题，另外恢复的效率也较低。

路由器对每个输出线路都有一个排队队列，当有线路空闲时，路由器就扫描队列，依次将每队等候的第一个数据包发出，形成公平排队算法。它的带宽分配独立于数据包大小，各种服务在队列中几乎是同时开始的。因此它在没有牺牲统计复用的情况下提供另外的公平性，与端到端的拥塞控制机制可以较好地协同，缺点是实现起来比较复杂。还有改进的算法，根据不同应用对带宽要求，每个排队队列采用加权方法分配资源，增加对不同应用的适应性。

· 源端从反馈中了解到网络的情况，随后将发出的数据包标记为可丢弃的数据包。它的优势是不需要超时重传，也不依赖于粗粒度的 TCP 定时，所以应用在对时延有一定要求的应用时效果较好。

· 为了避免丢弃属于同一连接的连续数据包，提高每个连接的吞吐量，可使用随机早期检测算法。它是按一定的概率丢弃进入路由器的数据包，通过分摊包丢失率，可以在各连接之间获得较好的公平性，对突发业务的适应性较强。它的问题是会引起网络的不稳定，在选择参数方面也比较困难。

在传统的 TCP 拥塞控制中，结合 IP 层的拥塞控制算法，将两者配合使用，能够较好地解决拥塞控制问题。

结束语 计算机网络在各领域获得了越来越广泛的应用，网络带宽和缓存等资源日益不够使用，拥塞问题还会继续。一方面要继续改进已有的端到端拥塞控制，将其作为网络中的主要拥塞控制机制。另一方面要利用价格等经济因素，限制用户对网络资源的不适当需求，阻止拥塞的发生。建立网络新的收费模型，用经济学理论分析具有拥塞特性网络资源的计费问题。还可以在路由器中采用包调度算法和缓存管理技术，在 IP 层实现拥塞控制的策略。TCP/IP 拥塞控制的设计和实现面临着许多两难的选择，在实践中往往只能折衷处理，例如在效率和性能之间的折衷，在实际中不可能找到一种设计，能够在所有环境中都是“最好的”。目前使用的拥塞控制方法和技术也面临着许多挑战，还有改进的余地。

参 考 文 献

1 Tanenbaum A S. Computer Network, Fourth Edition. Pearson Education North Asia Limied, 2003
2 胡金初，主编. 计算机网络，清华大学出版社，2004