计算机网络的拥塞控制

李保利　（北京九七一六信箱研究生科　100101）

摘要：本文在解释拥塞控制问题的基础上，重点介绍了 TCP 协议和 ATM 网络中的拥塞控制算法。

关键词：计算机网络　拥塞控制　TCP　ATM　Slow Start

一、基本概念

在计算机网络中，如果某一时刻网络负载超过了可用资源或者网络出现故障，那么拥塞就会发生。此时会有数据包丢失，传输延迟增大，网络吞吐量下降等现象。

直观上，解决网络拥塞应当从两方面入手：一是尽量避免拥塞发生；一是在拥塞发生时采取措施消除拥塞。前者是为了预防，力图使网络维持在最佳状态，避免拥塞；后者是为了补救，进行拥塞恢复。要真正避免网络拥塞是可能的，但这样做必然会牺牲网络的资源利用率。在追求公平、高速、高利用率的环境下，采用这种保守的方法显然是不适宜的。因此，简化拥塞避免算法着重拥塞出现时的恢复处理更为合理。

拥塞控制与流量控制密切相关。拥塞控制会抑制源方发送数据，起到流量控制的作用；相反，好的流量控制可以避免或推迟拥塞的发生。但是，拥塞控制与流量控制又存在差异。前者是一个关系网络全局的问题，它意在解决路由器等中间链路设备的瓶颈问题。它的规则具有“社会性”，要求做到公平合理。后者只涉及源方与目的方端到端的传输，是为了确保源方发送数据时不超过目的方的接收能力，解决目的方的瓶颈问题。这是二者的“个人”行为，不存在公平性问题。

在研究拥塞时有两个基本度量是应当加以考虑的——有用吞吐量（useful throughout）和有效延迟（effective delay）。有用吞吐量是最终应用可实际达到的吞吐量。例如如果在文件传输中丢失了一个分组，则此文件必须重传，有用吞吐量下降。有效延迟不是第一次不成功发送分组所产生的延迟，而是从第一次发送直到最后在目的方成功收到的整个时间间隔。图 1 描述了拥塞发生前后负载与网络性能（有用吞吐量、响应时间、有用吞吐量/响应时间）的关系。从图中可以看到，在拐点 knee 处网络开始出现拥塞，有用吞吐量增长速度下降，响应时间延长。随负载继续增加，网络达到拥塞崩溃点 cliff，有用吞吐量急剧下降，响应延迟迅速增大。在拐点 knee 处网络性能最佳。

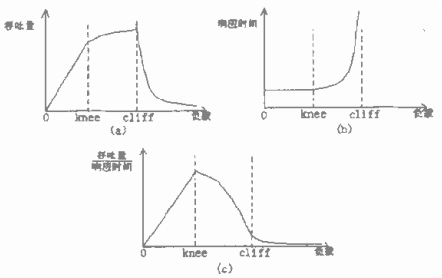

图 1　(a)拥塞发生前后负载与吞吐量的关系
(b)拥塞发生前后负载与响应时间的关系
(c)拥塞发生前后负载与网络性能（有用吞吐量/响应时间）的关系

二、拥塞控制的一般原则

网络出现拥塞意味着瞬时负载超过了系统中部分资源的处理能力。因此我们可以通过增加资源（能力）、降低负载来解决拥塞问题。比如，在拥塞出现时，可以使用备份的电话拨号线来暂时增加带宽；将传输分配在多个路由进行而不是总使用一个最好的路由也会显著增加传输带宽。此外，使用备份的空闲路由器也能在网络出现严重拥塞时增加网络带宽。

然而，如果有时不可能增加网络的处理能力或者这种增加已经达到极限，那么我们就只有靠降低负载来解决拥塞。此时，我们可以采取以下措施：拒绝向某些用户提供服务、降低对某些或所有用户的服务能力以及要求用户预先申请带宽等。

Yang 和 Reddy(1995) 依据控制论的理论将现有的拥塞控制算法分为两大类：开环控制和闭环控制。开环

控制不关心网络的当前状态，是预防性控制。它决定何时接受一个传输请求、何时丢弃数据包以及丢弃哪一个。闭环控制基于反馈的思想，它包括三部分：

1. 监控网络以确定拥塞发生的时间和场所。

2. 将以上拥塞信息(时间、场所等)传送给相应的设备以便它们采取纠正措施。

3. 相关设备通过调整网络的运行消除拥塞。

尽管在数据链路层、网络层、传输层应用的许多策略(如超时重传、失序缓存、路由选择等)都会影响拥塞的产生和控制拥塞，但明确的拥塞控制主要在网络层和传输层实现。

三、TCP 协议中的拥塞控制

当前 TCP 协议实现中的拥塞控制算法包括两部分：Slow Start 和 Fast Retransmit and Fast Recovery。它们是由 LBL(Lawvence Berkeley Laboratory)的 Van Jacobson 等人设计的。

1. Slow Start

Jacobson 和 Kerels 在 1988 年首次提出这个算法。

本算法的一个假设是网络故障导致的分组丢失非常少(少于 1%)。在这种情况下发生分组丢失就意味着源方与目的方之间出现了拥塞。分组丢失有两种表现：传输超时或收到重复的确认信号 ACK。

Slow Start 算法在源方为每个连接设置两个变量：一个为拥塞窗口，记为 cwnd；一个为 Slow Start 临界值，记为 ssthresh。记源方发送窗口为 swnd，目的方声明的接收窗口为 awnd。为简化描述，我们假定所有窗口大小以分组为单位计。整个 Slow Start 算法描述如下：

·对每个连接，初始时设置 cwnd = 1，ssthresh 取一个默认值。

·源方发送窗口 swnd = MAX(1, [MIN(cwnd, awnd)])

·当新数据被目的方确认时，源方就增加 cwnd 的值。增加的幅度与连接所处的状态有关。

```
if (cwnd < ssthresh) cwnd + = 1; /* 在 Slow Start 阶段或者说是在拥塞恢复阶段 */
else cwnd + = 1/cwnd; /* 在拥塞避免阶段 */
```

·当拥塞发生即传输超时或收到重复的 ACK 信号时，设置 ssthresh = [MAX(1, swnd/2)]。如果是传输超时，还要设置 cwnd = 1。

2. Fast Retransmit and Fast Recovery

由于目的方收到一个失序的分组时，它也发回一个重复的 ACK 信号。发送这一重复 ACK 信号的目的是要让源方知道目的方没有按次序收到分组，并告知期望收到的分组号。所以源方收到重复的 ACK 信号时并不能确定是丢失了分组还是发生了分组失序。通常假定，发生分组失序时，在目的方解决此问题之前，源方只可能收到一个或二个重复的 ACK 信号。如果源方连续收到三个以上的重复确认信号，就很可能表明一个分组丢失了。这时源方不等到重传超时就重新传送那个可能丢失的分组，这就叫做快速重传。

在快速重传可能丢失的分组之后，连接进入到拥塞避免阶段而不是进入 Slow Start 阶段。这就是快速恢复。实际上，既然目的方只有在收到另一个分组时才会发回重复的 ACK 信号，那么就表明此时源方与目的方之间仍存在着数据流，所以源方不必急剧降低数据流。这样做使大窗口下中度拥塞发生时网络仍具有较高的吞吐量。

快速重传和快速恢复通常一起实现，其算法描述如下：

·当源方连续收到第三个重复 ACK 信号时，设置 ssthresh = MAX(2, [cwnd/2])；重传丢失的分组，设置 cwnd = ssthresh + 3(目的方已经缓存了三个分组)。

·每次又一个重复 ACK 信号到来时，设置 cwnd + = 1。这是因为又一个分组到达了目的方。如果新的 cwnd 值允许，就传送另一个数据分组。

·当下一个确认收到新数据分组的 ACK 信号到来时，设置 cwnd = ssthresh(步骤 1 设置的值)。这个 ACK 信号是在一个回绕时间之后对步骤 1 中重传分组的确认。此外，这个 ACK 信号还确认了在丢失分组与第一次收到重复的 ACK 信号间发送的所有分组。

在 Jacobson 提出 Slow Start 算法之后，许多学者对这个 TCP 标准算法进行了研究。

Scott Shenker、Lixia Zhang 和 David D. Clark 等人利用模拟的方法研究 TCP 拥塞控制算法的行为，发现了两个意想不到的现象：其一，来自不同连接的分组，不是被混合在一起，而是被完全分割成不同的群(Cluster)；其二，在拥塞阶段，每一个连接丢失一个分组。另外，他们还定量分析了拥塞窗口 cwnd 随时间 t 的变化情况：先是迅速增长(log(cwnd) ct)，而后是线性增长(cwnd ct)，最后是缓慢增长(cwnd ct)(c、c、c 为常量)。在拥塞发生后，cwnd 急剧下降，然后重新开始以上增长过程。根据以上结果，他们提出了一些改进 TCP 拥塞控制算法的策略。比如，在拥塞阶段只允许一个连接丢失分组；在丢失

分组时，每个连接的拥塞窗口值不是急剧减小。

另外，Janey C. Hoe 通过观察 TCP 拥塞控制算法在起始(Start - up)阶段的表现，指出：选择合适的 ssthresh 值至关重要。小的 ssthresh 值，使拥塞窗口 cwnd 迅速地(指数型)达到 ssthresh，而后开始线性增长，这样能够逐步接近线路的传输能力而不至于过载(overfeeding)。但是，如果 ssthresh 过小，又会使拥塞窗口 cwnd 过早进入线性增长方式，延长数据传送时间，降低网络利用率。Janey 建议根据带宽延迟积(bandwidth - delay product)来设置和调整 ssthresh 值。此外，针对现有快速重传算法对单个分组丢失恢复的效果很好而对多个分组的丢失不够理想的情况，Janey 设计了一个改进的快速重传算法。

四、ATM 拥塞控制

拥塞控制是 ATM 网络传输控制与管理的核心问题。ATM 网络经许可控制建立连接后，基于连接的特性和 QoS 要求，必须给这个连接分配一定的带宽。由于 ATM 网允许所有的连接共享带宽资源，加上各类服务传输速率变化很大，因此，实际入网的传输流量很有可能超过分配给它的带宽，造成拥塞。这时就需要对传输流量进行监控，以保证传输流传输过程中的特性与它申请入网时要求的传输特性以及网络分配给它的带宽相符。

ATM 拥塞控制包括开环预防控制和闭环反馈控制。开环预防控制采用普通信元速率算法对信元的流速进行整形 (Traffic Shaping)。闭环反馈控制也叫反馈流控制。

1. 开环预防控制

(1)"漏斗"(Leaky Bucket)算法 目前最常用的带宽监控技术是普通信元速率算法，也叫"漏斗"(Leaky Bucket)算法，这种算法可将突发传输流转化为平缓传输流。漏斗算法用于确保用户的传输流遵守建立连接时的规定。其基本思想是：任何一个信元要进入网络，一定要从令牌池(漏斗)中取得一个令牌；如果此时令牌池为空，则该信元被丢失；令牌以网络平均允许速率 R 产生，令牌池最多可存放 M 个令牌(M 即漏斗的大小，由传输参数的突发容忍量表示)，令牌池满时，新产生的令牌被丢弃。

一个改进方法是在信元到达漏斗前增加一个缓冲区。这样，当令牌池为空时，只要缓冲器没有满，信元就可以缓存在缓冲器中而不被丢弃。由于这种改进以增加信元的等待时间来换取低的信元丢失率，因此选择一个合适的缓冲区大小以达到两者的最优控制是该方法的关键。漏斗算法也有一些缺点，例如即使在网络负载很低时，漏斗算法对违约信元仍然采用丢弃或放入缓冲区的方法，由于算法限制，减缓了传输流速，造成网络资源的浪费。采用标志法可以改善这一缺陷：当信元到达但令牌池为空或缓冲器已满时，就将该信元打上一个标志，说明是违约信元，然后允许它进入网络，如果在网络某处遇到拥塞，则丢弃；若一直没有遇到拥塞，则可到达目的结点。

(2)传输整形 (Traffic Shaping) 在 ATM 网络中，传输流是高度突发的，其传输速率变化很大，如果根据排队理论，适当地改善传输流进入网络的统计特性，使信元到达分布这一随机过程的统计特性越平滑，网络服务质量(延迟、丢失率)就越好。传输整形就是要避免信元在网络中的突发性传输，达到改善网络 QoS 性能的目的。漏斗算法注重带宽的增强，传输整形技术则注重于降低突发度和改进传输流进入网络的分布。传输整形技术成功的关键在减少了传输延迟和信元的丢失。

2. 反馈流控(Feedback Flow Control)

闭环、动态的流控需要反馈机制使信源知道拥塞点的信息状态。已被 ATM 网络研究的反馈流控方案主要有以下两类：

·基于信用的流控 (Credit - Based Flow Control) 突发传输造成网络超载的最显著的标志是缓冲溢出。信用流控就是直接控制缓冲的分配。在沿着链路向前传送任何信元前，发送结点首先需要接收来自接收结点的信用信息，以确定接收点是否有可接收信息的缓冲空间。发送结点接收到信用信息后，在接收结点可接收范围内传送一定数量的信元，从而保证不会发生拥塞。信用流控是虚连接(Virtual Connections)链路。它的特点是提高了链路的利用率、控制质量和公平性。最明显的一个缺点是它的复杂性和缺少灵活性，以致于这种控制方案没有被 ATM 论坛选用。

·基于速率的流控(Rate - Based Flow Control) 基于速率的流控直接控制连接的带宽(速率)。在 1994 年被 ATM 论坛选为可用位速率传输服务 ABR 的最佳控制方法。基于速率方案采用端 - 端控制，使用网络反馈信息规定每个虚连接上每个信源能发送的最大速率。基于速率流控方案包括速率的设置和速率的控制两个阶段。在这种方案中，分配给一个连接的带宽与两点间的延时无关，因此基于速率的流控在结构上是灵活的。这种方案支持公平的带宽分配，且这种方案无需复杂的队列管理。但是，它也面临着许多有待改进和研究的问题。

前向显式拥塞通告(FECN：Forward Explicit Conges-

tion Notification)采用前向显式拥塞指示(EFCI:: Explicit Forward Congestion Indication)机制。多个连接共享一个中间结点的链路队列,拥塞状态由中间结点链路队列的平均队列长度决定,当平均队列长度达到某一给定临界值时,通过队列的信元就打上一个 EFCI 标志,继续传输给目的结点,一旦目的结点接收到带 EFCI 标志的信元,就向信源发送拥塞信息 - - RM 信元(PTI = 110),信源据此按线性递增或倍减原则调整信元发送速率。由于速率的调整由网络拥塞状态和信元瞬时的传输速率共同决定,因此,该方案允许各个连接公平地分享带宽。其缺点是,拥塞信息传输路径过长,不利于信源对拥塞作出快速反应。

反向显式拥塞通告(BECN :Backward Explicit Congestion Notification)方案直接从拥塞点(如交换机)向每一个虚通道信源发送拥塞信息 - - RM 信元,因此,它的显著优点是对拥塞反应速度快。它对速率的控制简单、经济,因为它仅仅依赖于 BECN 信元来调整速率的变化。它的缺点是每一个中继结点都参与了拥塞控制,增加了系统开销。另外,如果交换机由于严重拥塞发不出 RM 信元,那么源方将会错误地增加速率。这是一种负反馈。

均衡速率控制算法(PRCA:Proportional Rate Control Algorithm) 源方发送的信元 EFCI! = 0,只有 n 个信元中的第一个信元的 EFCI 值为 0。目的方收到 EFCI = 0 的信元时发回一个 RM 信元。源方收到 RM 信元时提高发送速率(加增)。采用这种算法时.如果信元经过多个交换机,那么发生拥塞的可能性大,源方发送速率不高。

智能拥塞控制(ICCT: Intelligent Congestion Control Techniques)又称为增强型 PRCA 算法(EPRCA)。该算法的描述如下:

1. 源方发送信元时设置 EFCI = 0。每发送 n 个数据信元后,源方发送一个 RM 信元。这个 RM 信元中包括以下信息:当前速率 ACR(Current Allowed Cell Rate)、希望速率 ER(Desired Explict Rate)、拥塞指示位 CI(Congestion Indication Bit)。N 通常取为 30。

2. 源方初始时设置 ER 等于它的峰值速率 PCR (Peak Cell Rate),CI 位为 0。

3. 交换机从源方收到 RM 信元时,根据 ACR 计算平均允许速率 MACR(Mean Allowed Cell Rate)MACR = (1 - α) * MACR + α * ACR,其中 α 为比例因子,通常取 1/16。

4. 目的方收到 RM 信元时,根据最后一个数据信元中 EFCI 位的值设置 RM 信元中 CI 位的值,并将 ER 值设置为自己能够满足的大小,之后将 RM 信元发回源方。

5. 交换机收到目的方发回的 RM 信元时,根据自身的拥塞状态设置 CI 位的值(拥塞时 CI 设为 1)及 ER 的值。若当前缓冲队列长度大于某一个临界值,则 ER = MIN(ER, β * MACR)(为比例因子,通常取 7/8);否则 ER = MIN(ER, MACR)。

6. 源方在收到每个 RM 信元时,根据其中 CI 位的值调整发送速率。若 CI = 0,则 ACR = MIN(ACR + AIR, ER, PCR);若 CI = 1,则 ACR = MAX(ACR * RDF, MCR)。其中 MCR 为最小发送速率(Minimum Cell Rate), AIR 为 Addition Increase Rate, RDF 为 Reduction Factor。

另外,在 ATM 网络上传送 TCP、UDP 等数据分组时,由于一个数据分组要被分割成多个 ATM 信元来传输,所以如果丢失了一个信元,整个数据分组就要重传。这样网络拥塞时系统的有效吞吐量会很低。为解决这个问题,人们提出了两个简单的控制机制:部分分组丢弃(Partial Packet Discard)和早期分组丢弃(Early Packet Discard)。部分分组丢弃是指当丢弃一个信元时,ATM 交换机根据其后到来的 ATM 信元中 PTI 的 AUU 值丢弃同一个分组中的其他信元。早期分组丢弃要求设置一个临界值(Threshhold)。当交换机缓冲队列达到这个临界值时,就找到属于下一个分组的第一个信元,丢弃属于这个数据分组的所有信元。实验表明,早期分组丢弃的性能要优于部分分组丢弃。

参考文献

[1] Andrew S. Tanenbaum, Computer Networks (Third Edition), Prentice Hall, 1996

[2] V. Jacobson, Congestion Avoidanceand Control, Proceedings of SIGCOMM'88, August 1988

[3] Scott Shenker, Lixia Zhang, and David D. Clark, Some Observations on the Dynamics of a Congestion Control Algorithm., Computer Communications Review, Volume 20, Number 5 - October, 1990

[4] Janey C. Hoe, Improving the Start - up Behavior of a Congestion Control Scheme for TCP, Proceedings of SIGCOMM'96

(来稿时间:1998 年 7 月)