

组织：中国互动出版网 (<http://www.china-pub.com/>)

RFC 文档中文翻译计划 (<http://www.china-pub.com/compters/emook/aboutemook.htm>)

E-mail: [ouyang@china-pub.com](mailto:ouyang@china-pub.com)

译者：( )

译文发布时间：2001-12-28

版权：本中文翻译文档版权归中国互动出版网所有。可以用于非商业用途自由转载，但必须保留本文档的翻译及版权信息。

Network Working Group

John Nagle

Request For Comments: 896

6 January 1984

Ford Aerospace and Communications Corporation

## TCP/IP 互联网上的拥塞控制

(RFC896——Congestion Control in IP/TCP Internetworks)

这个文档讨论了 TCP/IP 互联网上拥塞控制的某些方面的问题。它旨在激发人们对这个问题的思考和进一步的讨论。为了实现改良的拥塞控制而提出某些具体建议时，这个文档并不具体制定任何标准。

### 引言

拥塞控制在复杂的网络中公认的问题。我们发现，国防部的网间网协议 (IP)，一种纯数据报协议，和传输控制协议 (TCP)，一种传输层协议，当把它们一起使用时容易遭受不寻常的拥塞问题，这是由在传输层和数据报层之间的相互作用而引起的。特别的，IP 网关对于被我们称为“拥塞崩溃”的现象而言是脆弱的，特别是当这种网关连到大范围的不同带宽的网络上的时候。我们研究了防止拥塞崩溃的方案。

由于这些协议在基于 ARPANET IMP 技术的网络上使用频繁，这些问题没有得到普遍的认识。基于 ARPANET IMP 的网络通常有一致的带宽和完全相同的交换节点，并且容量很大。对大多数 TCP/IP 主机和网络而言，盈余的容量以及 IMP 系统控制主机传输量的能力已足以处理拥塞。然而，随着最近 ARPANET 分成两个互连的网络以及连到 ARPANET 上的具有不同特性的其他网络的增长，IMP 系统良性特性中的可靠性已不足以允许主机迅速而可靠的通信。为了使网络成功的运转，必须改善拥塞控制。

福特航空航天及通信股份有限公司，和它的总公司，福特汽车公司，经营着如今实际存在的唯一一家私有的 TCP/IP 长距离网络。这个网络与四个网点相连（一

个在 Michigan, 两个在 California, 另一个在 England), 它们中的一些还有大规模的本地网。这个网络交叉连接在 ARPANET 上但却使用它自己的长距离线路。福特公司各网点之间通过私人租赁线路进行传输, 包括一条专用的横渡大西洋的卫星通讯线路。所有的交换节点都是没有点到点流量控制的纯 IP 报交换, 并且所有主机运行的软件都是由福特公司或它的子公司编写或者经他们大量修改的软件。这个网络上的链接带宽变化很大, 从 1200 到 10, 000, 000bps。通常, 我们已经没有能力购买昂贵的 ARPANET 那样的额外的长距离带宽, 而且我们的长距离链接在高峰时期是超负荷的。几秒的传输时间在我们的网络里是如此的平常。由于我们的纯数据报定向, 负荷过重和带宽的大范围变化, 我们不得不去解决 ARPANET/MILNET 组织才刚开始认识到的问题。我们的网络对主机的 TCP 实现的次最优性能很敏感, 包括与我们的网络连接或断开。我们力图检查在不同条件下的 TCP 性能, 并且已经解决了一些 TCP 普遍存在的问题。在这里我们提出了两个问题及其解决办法。许多 TCP 实现有这些问题; 如果对于某个给定的 TCP 实现, 经过 ARPANET/MILNET 网关的吞吐量比经过一个单一的网络糟, 那么很可能这个 TCP 实现存在这些问题中的一个或两个。

## 拥塞崩溃

在我们开始讨论这两个具体问题及其解决办法之前, 描述一下当这些问题没有解决时会发生什么是妥当的。在负载较重的带有端到端重发机制的纯数据报网络中, 当交换节点拥塞时, 网络上的往返时间增加, 在网络上传输的数据报的数量也增加了。这在轻负载下是正常的。只要在传输中仅有每个数据报的一个拷贝, 拥塞就在控制之中。一旦还没递送成功的数据报开始重传, 潜在的严重问题就可能会出现。

主机 TCP 的实现预期在增加的时间间隔内多次重传数据报, 直到重传间隔的某个时间上限已到。通常, 这个机制足以防止严重的拥塞问题。虽然有更好的自适应主机重传算法, 但是网络上的意外负载能使往返时间的增长速度比发送方估计的往返时间的更新更快。当一个新的数据报传输时, 这样的负载就产生了, 这样的文件传输开始填充一个大的窗口。如果这个往返时间超过了所有主机的最大重传间隔, 那么主机将开始向网络产生越来越多的同一数据报的副本。这时, 这个网络有了严重问题。最终在交换节点中所有可获得的缓冲将饱和, 于是必须丢失数据报。这时, 被传送的这个数据报的往返时间到达它的最大值。主机多次发送每个报文, 最终每个报文的某个拷贝到达它的目的地。这就是拥塞崩溃。

这个状态是稳定的。一旦到达饱和点, 如果选择包丢弃算法良好, 网络将继续运行在性能降低了的状态下。在这种状态下, 每个包被传输几次, 吞吐量将降低到正常情况的几分之一。我们实验性地迫使我们的网络处于这样的状态并且观察它的稳定性。往返时间可能变得很大以致于由于主机超时造成连接中断。

拥塞崩溃和不正常的拥塞通常不出现在 ARPANET/MILNET 系统中, 因为这些网络有足够大的超额容量。只要连接不经过 IP 网关, 增强的主机流量控制机制通常能防止拥塞崩溃, 特别是自从针对和纯 ARPANET 网情况相关的时间常量

很好地调整了 TCP 实现以来。然而，当 TCP 运行在 ARPANET/MILNET 上数据报在网关被丢弃时，除了 ICMP 的源抑制报文外，没有什么基本的机制来防止拥塞崩溃。一些运行不好的主机通过它们自身使网关拥塞并阻止其他主机通过是没价值的。我们已经在 ARPANET 上的某些主机上（我们已私下与这些主机的管理员交流过）重复观察到这个问题。

给网关添加额外的内存不能解决这个问题。添加的内存越多，在数据报被丢弃之前往返时间变得越长。这样，拥塞崩溃的发生将被延迟，但当崩溃发生时，网络中的更大的数据报分片将被复制，吞吐量将变得更糟。

## 两个问题

和 TCP 实现的技术有关的两个关键问题已经被观察到；我们称它们为短数据报问题和源抑制问题。第二个问题正由几个实现方案着手解决，第一个问题通常被（不正确地）认为已经被解决了。我们发现，一旦短数据报问题解决，源抑制问题就变得更加容易处理。因此，我们首先提出短数据报问题及其解决方案。

## 短数据报问题

这里有一个与短数据报相关的具体问题。当 TCP 用来传输来自键盘的单字符信息时，典型的结果是为了传输一个字节的有用数据传输了 41 个字节的数据报（1 个字节的数据，40 个字节的头文件）。这 4000% 的开销是令人讨厌的，但在轻负载的网络里是可以容忍的。然而，在负荷过重的网络中，由这个开销引起的拥塞能导致数据报的丢失和重传，以及在交换节点和网关中由于拥塞而造成传输时间过大。事实上，吞吐量可能降低以致于 TCP 连接被异常中断。

这个典型的问题在 20 世纪 60 年代下半期在 Tymnet 网络中第一次被提出并被广泛认识。那里所采用的解决办法是强行对每单位时间里所产生的数据报的数量给定一个限制。这个限制是通过短数据报延迟一个短的时间（200-500ms）后再传输来实施的，以期可以在定时器到时之前另一个或两个字符到来并附加在同一个数据报中。为了增加用户的可接受性，一个附加的特性是当一个控制字符（比如回车字符）到来时，禁止时间延迟。

这个技术已经在 NCP Telnet, X.25 PADs 和 TCP Telnet 中使用。它的优点是易于理解和不难实现。它的缺点是很难给出一个使每个人满意的时限。一个在 10M bps 以太网上提供高速应答服务的时限太短以致于不能防止在有 5 秒往返时间的高负荷的网络上的拥塞崩溃；相反，处理高负荷的网络的时限太长又会给以太网的用户造成挫折感。

## 短数据报的解决方案

显然，一个自适应的方法是不难想到的。我们期望为自适应交互包的时限提出一个建议方案，这个时限是在 TCP 所观察到的往返时间延迟的基础上的。然而虽然这样一个机制确实能被实现，但它是不必要的。我们发现了一个简单且优化的解决方案。

这个解决方案是，如果任何先前在连接上传输的数据仍没有被确认，那么来

自用户的输出数据到来时，禁止传输 TCP 数据段。这个限制是无条件的，没有定时器，不需要测试收到的数据的大小，不需要其他条件。典型的实现只需要 TCP 程序中的一两行程序。

乍看，这个解决方案好象意味着 TCP 行为的剧烈改变。但并非如此。最终它很好地工作。让我们看看为什么是这样。

当一个用户向一个 TCP 连接写数据时，TCP 收到一些数据。它可以保持这些数据以便以后传送或者也可以立即送出一个数据包。如果它不立即传送，它将在一个传入的数据包到来且改变了系统状态之后传送数据。可以有两种方式之一来改变这种状态；这个到来的数据包确认远端主机收到数据，或者通告远端主机为新数据提供的可用缓冲空间大小。（后者指“更新窗口”）。每次，此连接上的数据到来时，TCP 必须重新检查它的当前状态并可能会送出一些数据包。这样，当我们忽略送来自用户端的数据时，我们只是简单地延迟传输直到下一个来自远端的主机的报文到来。报文总是很快就到来，除非这条连接之前是空闲的或者与另一端的通信丢失。在第一种情况，即空闲连接，我们的方案将使用户在任何时候向 TCP 连接写数据时送出数据包。这样，我们就不会在空闲连接时死锁。第二种情况，远端主机失败，传送更多的数据都是无效的。注意，我们没有采取任何措施来禁止正常的 TCP 重传逻辑，因此，丢失报文不是问题。

这个方案在不同条件下的性能测试表明这个方案可以在所有情况下工作。第一个测试的情况是我们想解决面向字符的 Telnet 连接问题。让我们想象一下，用户每 200ms 向 TCP 输出一个新的字符，并且这个连接要经过以太网，此以太网的往返时间包括了 50ms 的软件处理时间。如果没有任何机制来防止短数据报的拥塞，与每个字符对应的数据包将被送出，响应是最佳的。开销将是 4000%，但在以太网上可以接受的。而典型的定时器方案，每秒两个数据包的限制，将使两个或三个字符在一个数据包中被传送。响应性能将降低，即使在高带宽的以太网上也是没有用的。开销降到 1500%，但在以太网上这是个不太好的替换方案。而我们的方案，用户所敲击的每个字符将找到一条空闲的 TCP 连接，字符将立刻被传送，正如在无控制的情况一样。用户将不会感觉到延迟。这样，我们的方案既可作为无控制的方案运行又可以提供比定时器方案更好的响应。

第二个测试的情况是同样的 Telnet 测试，但是测试是在有 5 秒往返时间的长距离连接上。如果没有任何机制防止短数据包拥塞，25 个数据包将在 5 秒内被送出。这儿开销是 4000%<sup>1</sup>。用典型的定时器方案且同样是每秒 2 个数据包的限制，将仍有 10 个数据包不能处理并引起拥塞。当然往返时间将不会因传送很多的数据包而得到改善；通常，它会因数据包竞争行时间而变得更糟。这时开销降到了 1500%。然而，用我们的方案，来自用户的第一个字符将发现空闲的 TCP 连接且立即传送。接下来的 24 个字符，以 200ms 的间隔从用户端到来，将等待远端主机来的报文。当一个对第一个数据报的确认 ACK 在 5 秒末到来时，这 24 个

---

<sup>1</sup> 在纯 ARPANET 网中没有这个问题，因为当没有处理的数据包过多时，IMP 机制将封锁主机，但在这种情况下，涉及到纯数据报本地网（比如以太网）或者一个纯数据报网关（比如 ARPANET/MILNET 网关），有大量的小数据报未处理是有可能的。

等待的字符封装到数据包中被传送出去。这样，我们的方案导致了在没有损失响应时间的情况下开销减少到 320%。用我们的方案响应时间通常因为数据包开销减少而得到改善。拥塞将减少且往返时间延迟将显著下降。在这个情况中，我们的方案明显优于其他方法。

我们对所有的 TCP 连接使用我们的方案，不仅仅是 Telnet 连接。让我们看看用我们的方法为文件传输建立数据连接时将会发生什么。我们将再一次考虑到这两个极端的情况。

象前面所提的一样，我们首先考虑以太网的情况。用户现在以 512 字节块的大小按 TCP 所能接受的速度向 TCP 写数据。用户第一次向 TCP 写的的数据启动传输；我们的第一个数据报的大小将是 512+40 字节或 552 字节。用户的第二次写数据不会引起传送但会使这个块被缓冲。设想用户在第一个确认到来之前填充 TCP 的输出缓冲区。然后，当 ACK 到来时，满足窗口大小的所有排队数据将被送出，从那时起，窗口保持为满，每个 ACK 开始一个传送循环，等待的数据被送出。这样，在一个往返时间初始周期以后，只有一个块要传送时，我们的方案解决了最大吞吐量的情况。由于启动延迟在以太网上只有 50ms，因此，启动的瞬时延迟是无关重要的。三种方案都对这种情况提供了等价的性能。

最后，让我们看看在有 5 秒往返时间的连接上的文件传输。在第一个确认回来之前只有一个数据包被发送；窗口将被填充并填满。因为往返时间是 5 秒，只有 512 字节的数据在第一个 5 秒被发送。假定有一个 2k 的窗口，一旦第一个确认来到，2K 的数据将被发送，之后，将保持每 5 秒 2K 的稳定速率。只有在这种情况下我们的方案才比定时器方案差，差别只在启动瞬时。稳定状态下的吞吐量是相同的。朴素的方案和定时器方案在上面所提的情况下都要花 250 秒来传输 100K 字节的文件，我们的方案将花 254 秒，1.6%的差别。

## 带 ICMP 的拥塞控制

解决了短数据报问题以及在我们自己网络中的极端的短数据报拥塞问题，我们把注意力转向通常的拥塞控制。既然我们的网络是没有点对点流量控制的纯数据报网络，对我们而言，在 IP 标准下可获得的唯一机制是 ICMP 源抑制报文。在精心的控制下，我们发现这足以防止严重的拥塞问题。我们发现留心主机或交换节点对源抑制报文的的行为是必要的。

### 何时发 ICMP 源抑制报文

当前的 ICMP 标准<sup>2</sup>规定，无论何时包丢失，ICMP 源抑制报文就应该被发送，此外，当网关发现自己资源不足时也应发送源抑制报文。这里有些二义性，但很明显，没有送 ICMP 报文而丢弃数据包违背了的标准。

---

<sup>2</sup> ARPANET RFC 792 是当前的标准。国防通讯部门通知我们，在 MIL-STD-1777 中的 ICMP 描述是不完全的，且在这个标准将来的修订版中将被删除。

我们的基本假设是，在网络正常运行时数据包不应该被丢弃。因此我们想在发送方使交换节点和网关超负荷之前抑制发送方重发。我们的交换节点在缓冲区耗尽之前能很好地发送 ICMP 源抑制报文；直到它们在发送 ICMP 源抑制报文之前必须丢弃报文时才等待。正如我们在短数据报问题的分析中演示的，仅提供大量的缓冲是不能解决问题的。通常，我们的经验是，当用了缓冲区的一半左右，源抑制报文就应该发送了；这不是基于广泛的实验，但似乎是个合理的技术决定。可以讨论一个自适应方案，这个方案可以调整基于近期经验的抑制产生边界；到目前为止我们还没有发现这个必要性。

还有其他的网关实现算法，仅在不只一个包被丢弃之后才产生源抑制报文。我们认为这种方法是不好的，因为任何基于包丢弃的拥塞控制系统浪费带宽，且可能在负荷重时容易受拥塞崩溃的影响。我们理解的是，被动地产生源抑制报文的方案基于这样的担心，害怕确认传输将被抑制以及这将导致连接失败。正如下面将要展示的，在主机实现中的恰当的源抑制控制排除了这种可能性。

## 在收到 ICMP 源抑制报文时做什么

当 ICMP 收到一个源抑制报文时我们通知 TCP 或者该层上的任意其他协议。我们的 TCP 具体实现的基本行为是减少与源抑制报文中所指出的主机的连接上未处理的数据的数量。使发送端 TCP 的反应就象远端主机窗口大小已经减少，从而实施这个控制。我们的第一个实现过于简单化但却有效；一旦收到源抑制报文，只要窗口不为空，我们的 TCP 就认为窗口大小为 0 并做相应处理。这个行为将持续到收到一定数量（现在是 10）的 ACK 为止，到那时，TCP 回到正常的运行状态<sup>3</sup>。Linkabit 公司的 David Mills 在他的 DCN 系统中实现了一个类似的但更详细的对未处理的数据包的节流控制。这个附加的复杂性好象提供了一个获得吞吐量的方式，但我们没有做过正式的测试。两个实现都有效地防止了交换节点的拥塞崩溃。

这样，源抑制方法有效地限制了到有限数量（可能为 1）的未处理报文的连接。因此，通信能继续但是速率降低，那正是期望的效果。

这个方案有个重要的性质，源抑制不能禁止确认和重传的发送。源抑制在 IP 层上的完全实现通常是不成功的，因为 IP 缺少足够的信息来正确地控制一个连接的流量。抑制确认信息往往产生重传和不必要的传输。抑制重传可能因重传超时而使连接丢失。我们的方案将在服务器超负荷下保持活动连接但减少每个连接的带宽。

在同一层与 TCP 一样的其他协议也应该响应源抑制。在每种情况下，我们建议应该控制新的传输流量但正常对待确认。唯一严重的问题来自于用户数据报协议，而通常不的主要传输发生器。我们仍未在这些协议中实现任何流量控制。

---

<sup>3</sup> 这点遵从控制学的观点“不要受比例控制的干扰除非它不工作了。”

## 网关的自防御

正如我们已经显示的，网关易受拥塞管理不善的主机的攻击。由于产生过多通信量而引起的主机错误行为不仅防止了主机自身的传输而且能影响了其他不相关的传输。这个问题可以在主机级处理，但既然一台有故障的主机能影响其他主机，将来的网关应该有防御能力使它们自己不被那些可恶可憎的主机的这种行为所影响。我们提供了一些基本的自防御技术。

曾经在 1983 年下半年，一个在 ARPANET 主机中的 TCP 故障使主机以 ARPANET 所能接受的速度疯狂地产生相同数据报的重发报文。通过 ARPANET 连接到我们网络的网关饱和，既然这个网关到 ARPANET 的带宽比到我们网络的要宽，少量有用的传输能通过。网关忙于发送源抑制报文但故障主机忽略它们。这持续了几个小时，直到故障主机崩溃。在这期间，我们的网络有效地从 ARPANET 上断开。

当网关被迫丢弃一个数据包时，网关慎重地选择要丢弃的数据包。做这个决定的典型技术是丢弃最近收到的数据包，或者数据包在最长输出队列的末端。我们提出一个值得的实用方法是，丢弃在网关中产生最多数据包的队列的对应主机所产生的最近的数据包，这种策略有助于平衡使用这个网关的各主机间的吞吐量。我们还没有尝试过这个策略，但它似乎是网关自保护的一个合理开始点。

另一个策略是丢弃新到来的数据包，如果这个数据包已经在队列中做了一个拷贝。如果使用哈希技术，为实现这一检查的计算负荷就不是问题。这个检查不能防止恶毒主机的攻击，但提供了一些保护措施来防止带低劣重传控制的 TCP 实现。如果本地主机与本地网络很好地协调工作，那么在快速本地网与慢速长距离网络之间的网关可以发现这个检查是有价值的。

理想的情况是网关应该检测出故障主机并抑制它们；这样的检测在纯数据报系统中是困难的。虽然，对 ICMP 源抑制报文的响应失败应该被认为是网关与主机断开的依据。检测这样的失效是重不寻常的但它是一个值得进一步研究的领域。

## 结论

与纯数据报网络相关的拥塞控制问题是困难的，但有效的解决办法是存在的。如果 TCP/IP 在重负荷下运行，TCP 实现算法必须以至少和这里描述过的解决方法一样有效的方式来解决这几个关键的问题。