DCCP 研究及其性能分析*)

郭晓峰　冯太明　周竞扬　陈贵海
(南京大学软件新技术国家重点实验室　南京210093)

DCCP Research and Analysis of its Performance

GUO Xiao-Feng　FENG Tai-Ming　ZHOU Jing-Yang　CHEN Gui-Hai
(State Key Lab for Novel Software Technology, Nanjing University, Nanjing 210093)

Abstract This paper explores simulation results of the Datagram Congestion Control Protocol(DCCP). The Internet community has seen the growth of real-time applications, which by the nature of their data flow do not desire the congestion control semantics of TCP and are thus left to build their applications on top of UDP. However, such methods have left with applications writers developing their own congestion control mechanisms. With congestion control mechanisms built directly into the transport layer, DCCP provides ease of use for application writers, as well as a selection of standards for use by applications. We explore the advantages of DCCP using an AIMD congestion control mechanism. An initial implementaion and experimentation of DCCP and its TCP-like congestion control mechanism show its behaviors.

Keywords DCCP, Datagram, TCP-like, AIMD, Congestion control

1. 介绍

随着 Internet 的迅猛发展，涌现了大量的实时应用，如 RealAudio、网络电话、在线游戏等等。它们对网络协议提出了新的要求。这些应用的特点是延时、抖动敏感，而对于一定数量的分组丢失则不会影响其服务质量。如实时应用存在回放点(play off)，晚于它到达客户端的数据将无任何意义。

实时应用的这种特点决定了能够确保数据正确传输的 TCP 并不适用，而 UDP 由于其轻负载、无连接、不可靠的特性，在这些应用中获得了广泛应用。然而，文[6]的分析告诉我们，长时间的、缺乏拥塞控制的 UDP 的大量使用，将会成为网络的隐患。在这样的情况下 DCCP(Datagram Congestion Control Protocol)应运而生。DCCP 提供一种拥有响应信息的不可靠的数据报流，便于与 UDP 的兼容。并且，DCCP 还提供了一个拥塞控制管理器。通过这个管理器，DCCP 的使用者可以选择自己需要使用的拥塞控制机制。这些拥塞控制机制用 CCID 来描述。目前 DCCP 协议中包含的 CCID 有两种：CCID 2，基于 TCP-like 的拥塞控制；CCID 3，基于 TFRC(TCP-Friendly Rate Control)的拥塞控制。

我们选用了 NS[18](Network Simulator)实现了 DCCP，并在此基础上对其进行性能评价。由于 DCCP 协议研究现今才刚刚起步，协议只拥有一个粗略的框架，在实现的过程中，还需要对协议理解后加入一些具体的实现算法。其中，我们设计了新的响应速率的计算方法，替代了协议中确定 AckRatio 的值的方法，获得了更快的速度。设计了将包的头中可变长度的部分移植到包体中的方法，避免了 NS 与实际实现协议的冲突，使应用与模拟的 NS 与实际协议的设计相适应。设计了自己判断接收包丢失的算法，避免将集体延迟的包当作丢失包对待。设计了判断最近一次响应窗口是否发送完毕的算法。另外，因为 NS 固有的特性，我们也对协议做了些改动。首先，我们削减了包的头类型，以符合模拟的需求。其次，我们去除了 DCCP 握手的过程，免除模拟不需要的内容。还有，我们将发送端和接收端分离设计，以清晰协议层次构造，方便模拟的进行。通过自己设计的模拟实验结果，和对实验结果的仔细分析，证实 DCCP 是一种可以避免与 TCP 争抢网络带宽的协议。

本文的第2部分将就当前实时网络传输中的一些其它方案进行介绍。第3部分将简要介绍 DCCP 协议。第4部分介绍了 DCCP CCID 2的设计与实现。第5部分分析 DCCP 传输的模拟结果。第6部分描述了未来还需要继续的工作。最后是小结。

2. 相关工作

自从1984年 Nagle 在文[1]中提出拥塞冲突的概念以来，网络中关于拥塞控制的研究就一直没有停止过。1986年 Jacobson 最早提出拥塞避免机制，并在其1988年的文[2]中详细讨论了这个方法。该文中提到的慢启动、快速重传和拥塞避免算法构成了现在常用的 AIMD(Additive Increase Multiplicative Decrease)机制的基础。1997年，这些算法合起来作为 RFC 公布[13]。在其基础上，后人提出了一些新的修改建议，主要改进同一个拥塞窗口中出现多个丢包情况的拥塞避免算法。这些算法典型的有 Reno TCP、New-Reno TCP、SACK TCP 等，文[3]通过模拟详细分析了这几种算法的性能。经过十几年的讨论和研究，TCP 上的拥塞避免机制获得了很大的改进，具体可见文[14～16]，其中详尽介绍了在这方面已经做过和即将要做的研究。

近年来，UDP 协议越来越广泛地应用在很多新的协议上。这些应用并不需要 TCP 中的可靠传输的性质，它们采用了 UDP。这些应用包括：Counter Strike[4]、Real Audio[5]以及

*)本文得到国家自然科学基金资助(项目编号：60073029)。郭晓峰　硕士研究生，主要从事网络拥塞控制、ad hoc 移动网络研究。冯太明　硕士研究生，主要从事 CORBA、网络 Qos 研究。陈贵海　教授，博士生导师，主要从事计算机理论、网络计算等方面研究。

IP Telephony 等等。由于 UDP 的一些缺点制约了这样的应用的发展，这些缺点包括：1）拥塞控制机制的实现比较困难；2）一旦拥塞出现的时候，UDP 常常出现异常。所以，这些应用常常需要自己去实现针对某种特殊情况需要的拥塞控制的机制。然而，拥塞控制机制的实现并不容易，尤其在应用层实现拥塞控制机制更为麻烦。因此，急需一种新的传输层协议在不可靠流的基础上完成拥塞控制的工作。

关于这方面工作的发展，主要有 RTP[11]（RealTime Transport Protocol）和 SCTP[12]（Stream Control Transmission Protocol），以及 CM[17]（Congestion Manager）。RTP 主要用于支持端对端的实时数据传输；SCTP 则提供了更好的可靠性，通过一些技术比如使用单一的连接来实现多应用和可调整的多条数据流；而 CM 则把拥塞控制的实现放到了更低的层次。

基于 UDP 之上的 RTP 应用可以简单地获得 RTP 的校验和和多路复用的服务。但是，RTP 并不能提供任何服务质量的支持，不能提供分时传输的支持。这样的一个协议并不能满足我们提供一种 UDP 替代方案的要求。这样的替代方案，要求协议可以支持一种标准的拥塞控制机制，方便应用层的使用。

SCTP 是 TCP 的一种替代方案，它提供了可靠的有序传输和多流支持。但是，在很多应用中，却并不需要 SCTP 所提供的可靠性。当然，我们可以修改 SCTP 中的 TCP-like 方案，这样就可以无序地发送数据了；同时，应用还可以使用它提供的拥塞控制机制。然而，由于效率问题（SCTP 开支巨大），SCTP 仍然不适用于现在使用 UDP 的流。因此，必须有一种新的传输层协议如 DCCP 来提供非可靠的传输和不同拥塞机制的选用。

Congestion Manager 则更把非可靠拥塞控制转移到了更低的层次。在这种模型中，应用层在 UDP 上提供序列号和拥塞控制的反馈，但是它需要传送反馈给处于 UDP 底下的 CM，该机制专门用来调节包的发送速率。但是这种方案碰到的问题并不比前面的少。它仍然为应用层设计者提供了不必要的障碍。使用它我们没法确定如何使用 ECN；放弃对不可靠流非常有用的 ECN 并不现实；而使用 ECN，又不可避免地要使应用层用户直接修改 IP 包头。最后，将拥塞控制设计在应用层，不可避免地降低了某些人通过更改代码逃避拥塞控制的难度。

它的改进办法很快又被提出，在文[17]中，Congestion Manager 被完全整合到 UDP 和 IP 层之间。因为端到端的拥塞控制机制不应该出现在路由器上，所以 CM 没有被包含到 IP 中。但是，在其上它又同时支持 TCP 和 UDP。所以，为了获取拥塞控制所需要的反馈信息，它需要额外的包头来携带序列号，除非这个头是一个 IP 选项（这样的方法很可能导致 IPv4 路由器出问题）。但是如果这样的话，它的出现将需要描述一种区别于 UDP 的不同的 IP 协议。因此，这个包看起来将不再像 UDP 包或者 TCP 包了。所以，它不可避免地要在某一个头信息中增加拥塞控制的域，而增加拥塞控制域浪费的包头，对一个新协议来说，是特别需要避免的。另外，使用这种方法，UDP socket API 同样要进行改变，不仅要确定发送什么内容，而且要提供不同的拥塞控制机制来满足应用需求。所以，实际上这种实现方法还不如设计一种新的传输层协议。

综上所述，这方面的工作不外乎三种方法：处于传输层之上的应用的开发者们在应用层中加入一定的机制，用这样的机制实现自己需要的拥塞控制；处于传输层之下的新的拥塞管理方法的设计；以及设计一种新的传输层协议来实现不可靠的拥塞控制。DCCP 采用了第三种方法，它起了一种桥梁的作用，使应用开发者们不需要自己去实现拥塞控制，而仅仅需要简单地从传输层中直接选择自己需要的一种机制；并且，它避免了在数据包中加入过多的控制信息，减少网络数据传送的冗余以及避免将网络层次结构复杂化。

3. DCCP 协议

3.1 协议概述

DCCP 的主要目的就是为了允许应用依照自己的需要选择自己使用的拥塞控制机制。现在 DCCP 提供了两种类型的拥塞控制机制：TCP-like AIMD 机制、Equation-Based TFRC 机制。在使用 DCCP 协议的连接中，可以依靠 CCID（Congestion Control Identifier）标识来指定使用哪种机制。

DCCP 有 9 种不同的包类型：DCP-Request，DCP-Response，DCP-Data，DCP-Ack，DCP-DataAck，DCP-CloseReq，DCP-Reset，DCP-Move。

DCCP 的包类型如此之多，有别于以往的协议比如 TCP、UDP 等。因此，也大大地提高了 DCCP 的灵活性与可扩展性。比如 DCP-Move 包就可以使用于移动设备上。

DCCP 的两端机器都可以建立自己的 Half-Connection（以下用 HC 代替）。每个 HC 表示本端往另一端发送数据，并接收另一端返回来的响应消息。使用这样的机制，可以方便上下行两端的速率不同的线路的使用。比如接收节目的时候，下行的线路需要使用较大的带宽，而上行线路只需要较小的带宽来提交请求，两端甚至可能使用不相同的拥塞控制机制。DCCP 的 HC 方案巧妙地解决了这一问题。DCCP 连接建立的时候，两端可以分别建立自己的 HC 与对方进行通信。当自己这一方的请求完成时，可以关闭自己的 HC，而只剩下接收对方信息的连接。只要两条 HC 没有全部关闭，DCCP 连接就仍然保持着。

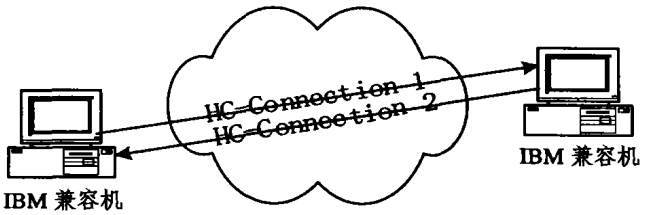

图1 DCCP 连接的基本结构

DCCP 中，使用了很多的 options 来交互两端的状态。特定的交互确定的连接属性称为特性（Feature）。Change、Prefer 和 Confirm 等 options 用来确定一些特定的特性的值。Ask、Choose 和 Answer 等 options 可以用来确定使用哪一些特性。重要的特性还有：CCID、TimeStamp、Ack Vector、Mobility、ECN Capable……。

3.2 CCID2和CCID3的描述

现在 DCCP 协议提供的 CCID 有两种：基于 TCP-like 的 CCID2以及基于 TFRC 的 CCID3。

TCP-like 拥塞控制机制是现在广泛使用的传输层的拥塞控制机制，经历了十几年的考验，性能稳定。它使数据流可以在端对端拥塞控制机制的限制范围内，最大限度地利用带宽。它与 TCP 使用的拥塞控制机制基本相似，而又略有不同。

它使用了经典的 AIMD 拥塞控制算法来调整流的发送速率。这样的控制机制在网络环境快速变化的情况下，可以尽可能快地获得相适应的带宽。DCCP CCID2使用它来实现与 TCP 协议最大限度的兼容。

TFRC 则是近年来被广泛研究的新型拥塞控制机制，它可以尽可能地减小发送端发送速率的突然性变化。DCCP CCID3实现适用于流媒体的非可靠数据报流的拥塞控制机制。TFRC 的 CCID3有助于需要 TCP-friendly 发送速率的应用程序，特别是在需要尽可能减小突然的速率变化的情况下。

当有新的拥塞控制机制出现的时候，DCCP 也可以很容易地将之归纳到现今的协议中去，通过增加新的拥塞控制描述符，来采用新的拥塞控制机制。

3.3 连接示例

本示例中，“发送端”表示 HC-Sender，“接收端”表示 HC-Receiver。

1）发送端发送 DCCP-Data 包，发送数据包的数目严格受拥塞窗口控制，就如 TCP 中一样。每个 DCCP-Data 包有一个顺序号。

2）每次收到发送端发来的含 Ack Ratio option 的包的时候，接收端马上发送一个 DCCP-Ack 包响应它。同样，每个 DCCP-Ack 包也拥有一个顺序号；并且，每个 DCCP-Ack 还要包含 Ack Vector 来告知发送端它的包丢失情况。

3）在拥塞窗口的控制下，发送方持续不断地发送 DCCP-Data 包。每次接到接收端发来的 DCCP-Ack 包，发送端检查其中的 Ack Vector Option，获知接收端收到和丢失的包的信息，据此调整自己的拥塞窗口大小。因为 DCCP 是不可靠的传输协议，所以发送端并不重新发送丢失的包。

4）因为 DCCP-Ack 包中也包含有顺序号，所以发送方可以根据 DCCP-Ack 包丢失的情况来调整 Ack Ratio。

5）在每个 Congestion Window 中，发送端至少发送一个 acknowledge 来响应接收到的接收端响应。这样的响应，发送端一般将其包含在自己的 DCCP-DataAck 包中。

6）发送端还要估计 RTT(Round-Trip Time)时间，并计算出 TO(TimeOut)值(DCCP 中 TimeOut 值的计算方法可以参考 TCP 中 RTO 值的计算方法)。这个 TO 值可以用来设置发送端的超时。当发送端超过 TO 时间没有收到响应的话，它将认为已经超时，据此调整它的拥塞窗口的大小和 Ack Ratio 的大小等。并重新发送新的包。

4. DCCP 协议的特性分析及实现改进

4.1 DCCP 协议的特性分析

首先，DCCP 的出现，就是为了避免 UDP 所造成的网络拥塞。在下文的性能比较中，我们可以看到，UDP 流的大量使用，将导致 TCP 流的瘫痪。而由于 UDP 流自身的不可靠特性，它也没法获得最好的性能，最终浪费网络带宽。DCCP 的出现，最大的好处就是可以避免这方面的问题。它与 TCP 同样通过接收端的反馈调整流的大小，两者通过合适的机制共享带宽。在尽力而为网络模型下，这样的方式有助于改善网络中流媒体的大量使用带来的网络带宽不足的问题。

其次，DCCP 是一种低开支、不可靠的拥塞控制协议。这样的特性，可以使它避免 TCP 流所必须要有的三次握手，它可以在每一次命令式的控制包中仍然包含数据信息。这样的特性，可以使它获取较低的时延。由于 DCCP 流头长度是可变的，其中最常用的仅包含数据的 DCCP-Data 型头只需要使用12个字节，虽然比 UDP 的8个字节稍大，却远小于 TCP 的20个字节。这样，在短包的传输上，比之 TCP 有先天的优越性。比如在电话应用中，每个数据流拥有带宽为20Kb/s，每个包发送时延为20ms，所以每个包大小约为50个字节。采用 TCP 的话，20字节被 IP 头占用，剩下30个字节给传输层的头和包内容；而如果采用的是 DCCP，12字节被占用，剩下38个字节。有效利用率是 TCP:UDP=30:38。而在移动网络中，带宽愈发地小，TCP 使用的局限也就更大了。

再次，DCCP 提供了更加灵活的拥塞控制选用机制。现在，DCCP 提供了 CCID 2和 CCID 3两种主要的拥塞控制机制，如果有新的需要，还可以不断地添加新的拥塞控制机制进去。使用时，仅需要在 DCCP 包的 Options 中选择相应的 CCID，既可以使用系统提供的拥塞控制机制。

另外，DCCP 还是用了一些还没有使用到 TCP 协议中去的一些拥塞控制研究的新进展。比如在 CCID2中，在一个拥塞窗口中，只接收 Ack Ratio 从接收端发来的反馈信息，而在每次的反馈信息中，包含这之前没处理的所有的包信息。还有，DCCP 协议是数据包传输协议，所以在拥塞窗口的计量单位是基于包的，而不是 TCP 协议中所用的字节。

4.2 DCCP 协议的一些实现改进

1）AR 的计算公式　在 CCID2中，Ack Ratio(响应频率)的出现是一个比较大的改进，它抑制了接收端过多的响应包的发送。由于采用 Ack Vector 来传送包的丢失情况，所以响应包有可能远大于 TCP 协议中的响应包。Ack Ratio 的使用首先有利于减小响应包对网络资源的占用。另外，响应包的减少，在慢启动的阶段，相应地会减缓发送端拥塞窗口的增长速度，避免拥塞窗口仅以指数级速度增长，影响对发送速率的准确定位。在 CCID2的描述中，它要求当收到的 ack 超过 K 个窗口的时候，相应地减小 Ack Ratio，其中 K=W/(R ^ 2-R)。但是当0<K<<1而 Ack Ratio 比较大时，收到一个 ack 的时候，就有可能是好几个 K 窗口的大小。所以，我设计了新的 AR 计算方法：

$$r1=\frac{ack\_count\times R\times R}{cwnd\times cwnd+ack\_count\times R}$$

$$R=R-r1(R-AckRatio)$$

这样的算法，可以避免当 K→0时，要通过无数次的 AckRatio-的计算才能确定新的 AckRatio 的问题。

2）丢失包的判定方法　在 DCCP 中，接收方不采用 TCP 协议中的每收到一个包即发送一个返回包的方法，而是在收到第 i 个响应包的时候，再由接收方同时发送前面收到的 n 个未被确认的包的响应包。前面 n 个包的响应信息集成到 AckVector 中，由接收方采用自己的方法判定是否有包丢失。因为在 DCCP 协议的描述中并没有具体化如何确定包的丢失，所以在我的算法中，我这样定义丢失包：

$\forall x, \exists\ 0<a_1<a_2<\cdots<a_{NUMDUPACK}$，

使得$\forall i\in[1\cdots NUMDUPPACK]$，第 $x+a_i$ 个包确认收到，则第 x 个包确认丢失。

3）判断最近一个拥塞窗口大小的包是否发送完毕的算法　由于 DCCP 中采用了 AckVector 的机制(如上所述)，所以发送方必须在接收到一定数量的接收方发来的响应包时，发送这些响应包的响应给接收方，以减少接收方发送的 AckVector 中包含的确认接收信息包的数量。在 DCCP 协议中，并没有规定如何判定何时传输完一个拥塞窗口数量的包，特别是当拥塞窗口根据反馈不停变化的情况下，所以我采用

了以下算法解决这个问题。基本的想法就是每次依据反馈信息更改拥塞窗口，拥塞窗口更改的同时不改变接收反馈包的累计情况，最后使用新确定的拥塞窗口大小来对反馈进行处理，具体请见下列算法描述。

收到接收方发来的响应包时，发送方必须完成以下任务：

i. 依照对方发来的 AckVector，找出最大的 x，使得∀ y< x，第 y 个包在 AckVector 中被接收方确认接收。

ii. 求出 x 对应的发送方发送的包的编号，并求出该编号与发送方最近一次发送 Ack of ack 中确认的编号的差值，设为 ack_count。

iii. 依据响应信息更改 Congestion Window 的大小，设为 cwnd。

iv. 依据 ack_count 和 cwnd 的大小，决定是否发送“ack of ack”。

i) 若 ack_count > cwnd，则发送“ack of ack”，并且标记最后一次发送的确认编号。

ii) 若 ack_count < cwnd，则继续。

5. DCCP 协议实现及性能分析

5.1 NS 中 DCCP 的实现

NS(Network Simulator)是一个用来进行网络研究的离散时间驱动的模拟器。它提供了对于有线、无线等多种环境下的 TCP、路由、多播的支持，为网络研究提供了方便。NS 最早出现在1989年 Berkeley 的 REAL Network Simulator 项目中。1995年，它获得 DARPA 的支持，正式成为网络研究最权威的模拟器之一，为网络研究的优劣提供了一个很好的评判标准。因此，我们采用了 NS 作为我们评价 DCCP 性能的工具。

NS 中的单向 TCP 实现中，一般采用 TCPAgent 来发送 TCP 包，而用 TCPSinkAgent 接收。所以，在我们的 DCCP 实现中，借鉴了 TCP 的做法，采用 DCPAgent 类来发送 DCP 包，而用 DCPSinkAgent 来接收 DCP 包。这些类的层次结构如图2。

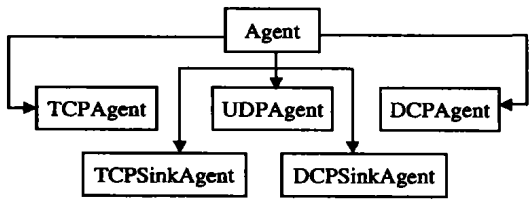

图2 NS 中 DCCP 实现层次结构

5.2 DCCP 性能分析

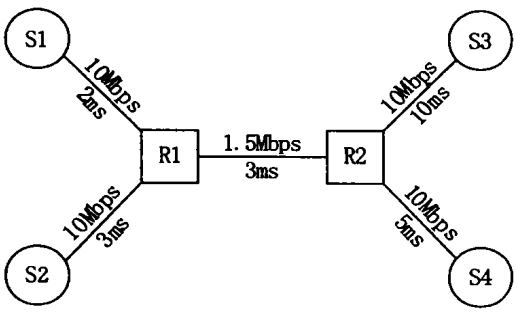

图3 模拟网络拓扑图

图3的拓扑结构取材于文[6]，是 UDP 与 TCP 争抢带宽问题的一个典型网络拓扑结构图。当没有拥塞控制机制的 UDP 流出现，与 TCP 流争抢 R1-R2带宽时。TCP 的发送端会减小自己的发送速率，绝大多数的带宽将会被 UDP 占用，尽管 UDP 仍然丢失了很多的包。

图4描述了这一典型问题(该图取自文[6]的 simulator script 的结果)。本图中，TCP 连接有3个，都从 S1→S3；每个连接都有无限的数据要传送。UDP 连接有一条：S2→S4；该连接的传送速率为常数。X 轴表示 UDP 包的发送速率。“Goodput”表示一个流占用的带宽，不包括重复发送的包。实线表示 UDP 发送的速率，虚线表示 UDP 占用的带宽，点线表示 TCP 占用的带宽。

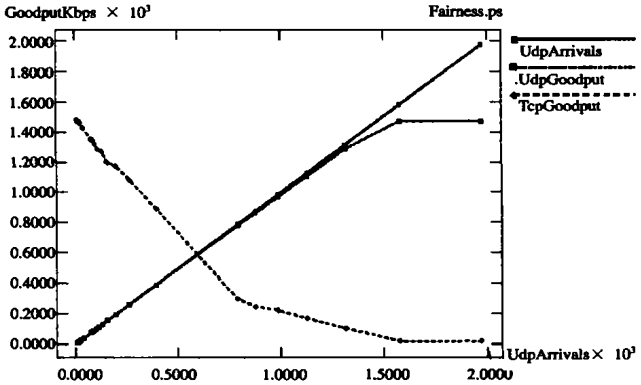

图4 3个 TCP 流和一个 UDP 对线路的占用状况分析

由图4可见，当 UDP 的发送速率较小时，带宽几乎全部为3个 TCP 连接所占用。当 UDP 发送速率变大时，带宽全部为 UDP 抢用，TCP 丢失了全部的带宽。不可靠流(无拥塞控制机制)和可靠流对带宽的争抢造成了这种情况；另外一个方面，这种情况也源自于 FIFO 型的路由器排队算法(本示例中，图4与图5均由 NS 自动生成)。

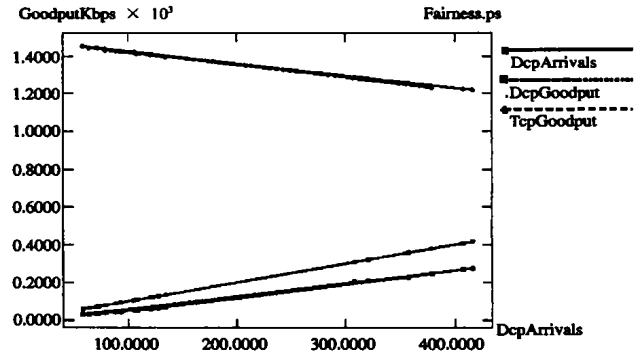

图5 3个 TCP 流和一个 UDP 流对线路的占用情况分析

在图5中，我们采用了 DCP 代替 UDP。同样的，我们有3条 TCP 连结和1条 DCP 连接。X 轴表示 DCP 包发送的速率，其他以此类推。由图可见，DCP 在需要带宽小于1/3的 TCP 总带宽的时候，可以发送出自己需要发送的包。而由于此时与 TCP 包的冲突，开始时有部分包不能成功地发送到接收端(与 TCP 流互相退避，空出带宽的阶段)，最后占用的带宽小于请求的带宽。而当发送端请求发送超过1/3总带宽时候，由于拥塞控制机制的抑制，这些包没能在发送端发送出去，避免了网络的拥塞。这一点可以从当发送端应用 CBR 发送速率大于1/3TCP 时，DCP 发送端速率一直保持不变看出。

通过上面的性能分析图，我们可以发现 DCCP 是一种可以在网络中和 TCP 共享带宽的协议。一旦网络发生拥塞，

DCCP 流将和 TCP 流依照各自的发送速率分配带宽，而不是各自抢占带宽。这种方式极高地提高了网络资源的使用效率。随着更多方式的拥塞控制机制加入到 DCCP 协议中，拥塞控制更广泛地引入非可靠传输协议中，网络中 UDP 对网络资源的争抢造成的不公平将成为过去。

6. 以后的工作

在 DCCP 的实现方面，仍然有很多工作需要我们去完成。现在我们只是完成了 NS 中的 DCCP 的 CCID2的实现。在现有的 DCCP 的协议描述中，还有 CCID3需要实现在 NS 中。当两种实现全部完成的时候，就可以在不同的 HC 上选用不同的拥塞控制方法，测试更多的 DCCP 的性能。

DCCP 中提供的特性可以方便加入需要的特性。而新的特性的发现与加入，也需要更多的研究和探索。比如如何在 DCCP 中扩展其移动性能，如何在 DCCP 中加入安全性能。

拥有拥塞控制机制的不可靠传输协议的出现，基本上可以解决网络中不可靠流抢夺可靠流带宽的问题，只要把不可靠流使用的协议由 UDP 换成 DCCP 即可。但是，在应用不广泛、使用不方便等等情况的阻碍下，在以后一段时间，绝大多数人使用的不可靠传输协议仍然会是 UDP。因此，当不可靠流和可靠流争抢带宽的情况发生时，如何让路由器发挥出自己的作用也值得我们去研究。

结论 DCCP 为现在使用 UDP 作为传输层协议的应用提供了一种替代方案。实际上，它综合了 TCP 和 UDP 两者的优点，利用了 TCP 二十年来在拥塞控制方面的研究成果。我们的研究充分证明了这个协议在性能方面可以避免 UDP 所造成的争抢网络资源的现象，显示了 DCCP 存在的意义和它替代 UDP 成为不可靠传输层协议的潜力。

致谢 感谢陈贵海老师对本论文的指导。感谢南京大学分布式实验室在实验环境方面的支持。感谢 DCCP maillist 上对我提出问题进行中肯解答的 E. Kohler 先生。

参 考 文 献

1 Nagle J. Congestion Control in IP/TCP. RFC896, Jan. 1984
2 Jacobson V. Congestion Avoidance and Control, ACM SIGCOMM '88, Aug. 1988
3 Fall K, Floyd S. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP
4 Counter Strike. http://www.counter-strike.net
5 RealAudio Player. http://www.real.com
6 Floyd S, Fall K. Promoting the Use of End-to-End Congestion Control in the Internet. IEEE/ACM Transaction on Networking, Aug. 1999
7 Kohler E, Handley M, Floyd S, Padhye J. Datagram Control Protocol. http://www.icir.org/kohler/dcp, Nov. 2001
8 Floyd S, Kohler E. Profile of Congestion Control ID 2: TCP-like Congestion Control. http://www.icir.org/kohler/dcp, May 2002
9 Padhye J, Floyd S, Kohler E. Profile of Congestion Control ID 3: TFRC Congestion Control. http://www.icir.org/kohler/dcp, June 2002
10 Floyd S, Handley M, Kohler E. Problemtatement for DCP. http://www.icir.org/kohler/dcp, Aug. 2002
11 Schulzrinne H, Casner S, Frederick R, Jacobson V. RTP: A Transport Protocol for Real-Time Applications. RFC1889
12 Stewart R, Xie Q, Morneault K, et al. Stream Control Transmission Protocol. RFC2960
13 Stevens W. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. RFC2001, Jan. 1997
14 Braden B, et al. Recommendations on Queue Management and Congestion Avoidance in the Internet. RFC2309, April 1998
15 Floyd S. A Report on Some Recent Developments in TCP Congestion Control. IEEE Communications Magazine, April 2001
16 Floyd S. Congestion Control Principles. RFC 2914, Sep. 2000
17 Balakrishnan H, Rahul H, Seshan S. An Integrated Congestion Management Architecture for Internet Hosts, ACM SIGCOMM, Sept. 1999
18 NS-2 simulator. http://www.isi.edu/nsnam/ns

（上接第80页）

中对协议模型的描述，可以将 TAO 的可插入式协议框架进行抽象和推广，得出图4(b)所示的通信协议模型。

小结 本文系统而详细地阐明了开放系统中实时 CORBA 技术的规范、实时 CORBA 的关键技术，为使用实时中间件实现开放系统的实时扩展提供了切实可行的方法；同时文中也启示我们两点：一、由于存在实时 CORBA1.0规范，当今研发出的可实用的实时 CORBA 系统大多使用优先级静态调度的机制，这与更为理想的动态调度尚有较大的差距。二、实时 CORBA 技术的实现不能脱离实时操作系统和实时网络的支持，但现有的实时网络协议（尤其是 TCP/IP 协议上的实时）的研究和实现与理想目标尚远，所以，要真正依靠实时中间件来构造实时开放系统，还有较多的研究工作要做。以上这些研究正是我们应该继续努力的方向。

参 考 文 献

1 骆志刚，胡健，刘锦德. 开放系统中的实时性. 计算机科学，2001(1)
2 Fay-Wolfe V, et al., Real -Time CORBA. IEEE Trans. on Parallel and Distributed Systems, 2000, 11(10)
3 刘锦德. 对于开放系统内涵的澄清. 计算机应用，1997. 6
4 Zhong Deng, Liu J W-S, Zhang L, Seri M, Frei A. An Open Environment for Real-Time Applications. Real-Time Systems Journal, 1998
5 Schmidt D C. An Overview of the Real-time CORBA Specification. IEEE Computer special issue on Object-Orient Real-Time Distributed Computing, 2000, 33(6)
6 OMG. Real- Time CORBA1.0 Adopted Specification, 1999
7 Pyarali I, et al. Applying Optimization Principle Patterns to Real-Time ORBs. the 5th USENIX COOTS'99, 1999. 5
8 OMG. The Common Object Request Broker: Architecture and Specification, OMG Document, 2.4.2, 2000.5
9 Schmidt D C. A High-performance Endsystem Architecture for Real-Time CORBA. IEEE Communications Magazine, 1997, 14(2)
10 DARPA. The Quorum Program. http: // www.darpa.mil/ito/research/quorum/ index.html. 1999
11 O'Ryan C, et al. The Design and Performance of a Pluggable Protocols Framework for Real-time Distributed Object Computing Middleware. In: Proc. of IFIP/ACM Middleware 2000 Conference, Pallisades, New York, April 2000
12 Crane J. Dynamic Binding for Distributed Systems: [Ph. D thesis]. University of London, May 1997