UDT协议及其拥塞控制机制

[朱先飞 王磊]

摘要 介绍了一种基于UDP的高性能数据传输协议UDT，并分析了UDT的拥塞控制机制。 UDT是面向连接的流数据传输协议，结合了窗口流量拥塞控制和速率控制两种控制方法，采用了新的DAIMD速率控制算法和带宽估计算法，支持高速广域网上的海量数据快速传输。

关键词： UDP TCP UDT AIMD DAIMD 拥塞控制

朱先飞

男，1998年毕业于长沙铁道学院计算机系，本科，现工作于中国电信股份有限公司广州研究院，从事软件研发、宽带差异化、媒体通信网络方面的工作

王磊

男，2004年毕业于西安电子科技大学，本科。现工作于中国电信股份有限公司广州研究院，从事宽带差异化、宽带智能终端的研究工作。

引言

传输控制协议（TCP）已经在目前的网络中取得了巨大的成功，然而，TCP协议并不是完美的，TCP本身针对拥塞问题的 AIMD(additive increase multiplicative decrease)算法无法很好地在高带宽网络环境中充分利用带宽，同时它在数据包的修复上也一直存在性能问题，TCP协议的缺陷在高速网络中正日显突出，而且应用也对新的协议有了需求。被多方认知的新传输协议有 Fast TCP、BiC TCP、 Scalable TCP、HighSpeed TCP、XCP等，但它们大多仅能在模拟实验中表现卓越，要将它们应用在实时的大范围网络环境中依然存在各种问题。因此，人们考虑从协议的应用层面着手，以 UDP为基础发展传输协议，比如 UDT、SABUL、Tsunami、RBUDP、FOBS以及 GTP，这类基于 UDP的传输协议往往具有轻便、容易安装、调试时间少、测试周期短等优点。

本文主要介绍UDT及其拥塞控制机制。

1 UDT简介

UDT是 UDP-based Data Transfer Protocol的简称，即基于 UDP的数据传输协议，是一种网际网络数据传输协议。UDT是开源软件，UDT的主要目的就是针对“TCP在高带宽长距离网络上的传输性能差”这一问题，尽可能地全面支持高速广域网上的海量数据传输。UDT建于UDP之上，引入了新的拥塞控制算法和数据可靠性控制机制。UDT是完全在 UDP上实现的，面向双向的应用层协议，因此它不仅支持可靠的数据流传输和部分可靠的数据报传输，也可以应用在除高速数据传输之外的其它领域，例如点对点技术，防火墙穿透，多媒体数据传输等。

2 UDT的设计动机和理念

尽管高效的数据传输一直是网络传输协议的设计目标，但数据传输效率往往会随着BDP的增长而下降。BDP，全称bandwidth delay product，即频宽－延迟乘积，它由网络连接容量（link capacity）和往返时间（RTT, Round-Trip Time）共同决定。在网络传输中，数据传输的友好性和稳定性有极其重要的地位，因此，在对数据传输高效率的界定与实现上会受到相当的束缚。目前大多数被广泛使用的传输协议都是被设计为在以 bytes 每秒的环境下运行的，均没有在高 BDP环境下进行过测试，对于高带宽环境下的应用效果仍有各种保留。对于离散系统下的数据密集型应用来说，它们无法容忍旧型传输协议所允许的延迟。其对每一个事件的执行大都设有一个“最大时间限制”，应用必须在这一段时间内完成任务，否则效率不足的实行将会引起数据包的丢失与超时报告。它们对数据传输协议可提供的传输速度有时甚至可高达“100000个数据包每秒”这样的要求。然而，在实际中要达到这种程度并不容易。从理论上说，若能在传送层或更低级的基础层面实现将是最理想的，但在现实中并不实际，反而更多地会选择在应用层上实现。而传送层上的UDP用户数据报协议正好提供了一个可在其基础上进行扩展的空间。

3 UDT的拥塞控制

（1）应答

应答是拥塞控制中是不可缺少的一环，但在高速网络中，生成和处理应答包的时间代价并不可忽略，应答包自身有一定大小，会占用一定的带宽。因此 UDT选中了以时间为基准的选择性应答。UDT会根据一个固定的时间差生成应答包，即传输的速度越快，控制阻塞所消耗带宽的比例就越小，固定的间隔与速率控制的间隔保持一致。不过，若在低带宽的状态下，UDT传输协议将会以累积确认协议（cumulative acknowledgement）实行应答机制。为了实现这个模型，NAK（negative acknowledgeement）被用作于报告包，用于传递在传送中丢失的信息，并激活拥塞控制运行机制。当包丢失被探测出来，NAK随即被生成并被传送到数据源。包含丢失包序列号的信息将在间隔被延长后被发送出去，这样即表示超时需重发，或NAK本身已经丢失。

（2）窗口流量控制与速率控制

在 UDT中，窗口流量控制算法并不会将数据以包的形式单独发送，而是将大批量的包数据一同发送出去。这种高密度的信息传输很容易形成信息爆炸，从而导致当发送者获悉问题时已经造成拥塞。与此同时，为了减缓这种信息爆炸的后遗症，普遍的做法是在路由器上设置一个与 BDP具有相同大小的缓冲区。但是，在高 BDP的环境下，这种做法显然是不可行的。针对在 TCP运用的 TCP pacing（在预设范围为cwnd的拥塞窗口内以一平均间隔传送包），为了不减少最后的网络吞吐量，在高 BDP连接下运用速率控制（Rate Control，RC控制）机制来直接制定包发送的间隔成为了UDT所选择解决此问题的做法。

只使用窗口流量控制已不能满足当前的网络需求。同样，若只使用 RC控制，一样无法达到期望的结果。例如，在只使用 RC控制的前提下，若网络连接中已出现了拥塞，但数据源依旧不断地发送数据直到它收到关于数据丢失的报告或者超时信息，这将导致大量的数据丢失。因此，需要有一个特定的控制算法来定义一个阈值来划分已应答和未应答的包。而这个现成的控制算法，即是窗口流量控制。

UDT以 RC控制为主，窗口流量控制为辅，共同组建成其自身的拥塞控制算法。UDT用 RC控制算法确定传送周期，用窗口流量控制算法判别包的应答状况。值得补充的是，UDT的窗口流量控制亦可称作流控制，它通过反馈在拥塞窗口范围和当前可用的接收端的缓冲区大小之间的最小值，来合并为一个简单的流控制结构。拥塞窗口的大小W根据公式保持动态更新，W = AS * (SYN + RTT)，AS是包抵达时的速度，通过计算包到达中间节点时的间隔而得出的。而在整个UDT传输协议中，SYN值为每次应答的间隔时间。因此对于每个包应答的过程来说，数据包传送的最大值等于接收端接收时的传送速度乘以RTT与SYN的和 (RTT+SYN)。这也获得了窗口流量大小的即时动态值。

（3）DAIMD

TCP中使用的 AIMD速率控制算法是兼顾稳定性和公平性的，但其每次RTT带来的网络开销无法满足高 BDP连接的需求。因此，AIMD算法在UDT中被重新定义，发展成为 DAIMD算法，即递减的 AIMD 速率控制算法。对所有速率控制间隔来说，如果没有NAK从接收端发出，而是发送ACK，包的发送速率增加为：x←x+α(x)（α(x)为非增长函数且随 x的增长而无限接近零）。如果有NAK发出，包地发送速率减少为：$x \leftarrow (1-\beta) \cdot x$（β为固定系数，β(0<β<1)）。这样当更改a(x)的值时，便可得到一组附加参数越来越小的RC控制算法值，即DAIMD算法结果。若将速率控制间隔作为一个单位时间，从时间点t到时间点t+1，发送率的增幅是：$x(t+1)=x(t)+\alpha(x(t))$，减幅是：$x(t+1)=(1-\beta)^n \cdot x(t)$（n为NAK的次数）。假设p(i)为i号包在传送过程中丢失的机率，D为双向延迟的时间值，那x(t D)则为在时间$(t, t+1)$内可能反馈的包数量，而$P(i) \cdot (1-(1-\beta)^i)$则是丢失可导致的减幅。

$$x(t+1)-x(t)=P(0)\cdot\alpha(x(t))-\sum_{i=1}^{x(t-D)}(P(i)\cdot(1-(1-\beta)^i))\cdot x(t)$$

将$p(0)$和$p(1)$看成独立事件且 $p(0)+p(1)=1$，以此为前提化简上面的公式再求导，经过一系列的计算及再化简，可得出的 UDT拥塞算法的实用程序函数

$$U(x)=\int\frac{\alpha(x)}{x\cdot\alpha(x)+\beta\cdot x^2}dx$$

由数学的函数特征来分析，证明DAIMD算法的趋向稳定及相对平衡是十分容易的，只要对实用程序函数进行二次求导便可。

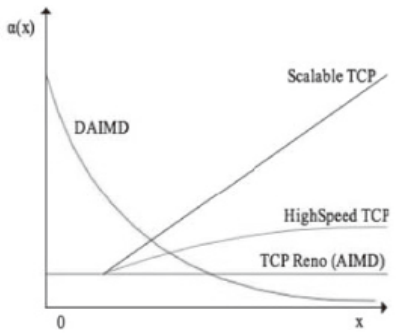

图1

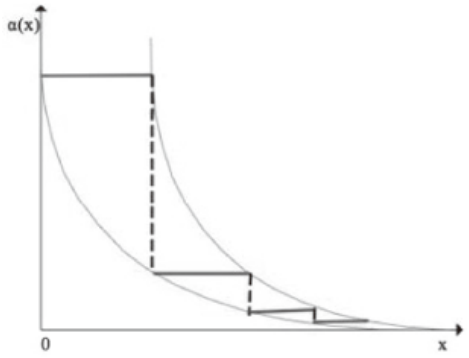

图2

图1：随着窗口大小的增加，TCP类协议中的 a(x)均在增长，而运用 DAIMD的 UDT却是在减小。

图2：此图展示DAIMD流可能首先能探查到的可用带宽，而每一阶段的线段长度便是其在此阶段上俱有的积极性。线段越长，UDT在此阶段表现得更积极。

UDT的速率控制是一个特定的 DAIMD算法，它对于a(x)有其独立的定义：

$$\alpha(x)=10^{\lceil\log(L-C(x))\rceil-\tau}\times\frac{1500}{S}\cdot\frac{1}{SYN}$$

x的单位是包/秒，L是连接极值单位为bit/秒，S是大小单位为Bytes的UDT包。C(x)是用于将现在传送率 x从单位包/秒转换到bit/秒的公式。C(x) = x * S * 8。τ是协议参数，在 UDT中为 9。公式中的1500/S是为了平衡各个包之间大小差异而定义的一个平衡系数（UDT将 1500bytes定义为标准包的大小）。

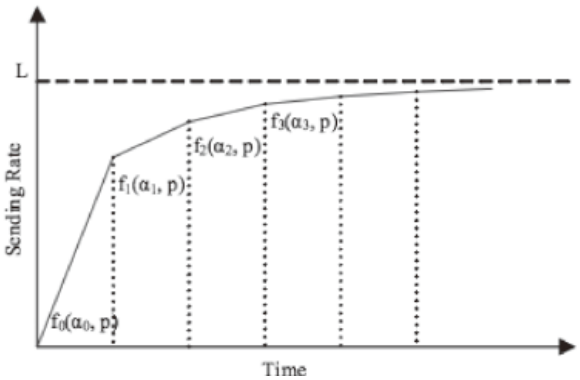

图3

由于 C(x)公式的存在，UDT的拥塞算法有几个阶段。如图3所示，UDT在初期，传动率攀升的速度十分之快，但随之越接近连接的容限极值L，开始慢慢减速。另

外，除了第一阶段外可能会有所不同外，UDT中其余每个阶段的时间范围都是完全一致的。

每次SYN间隔时间，如果上一次SYN时间没有NAK，但有ACK被接收，那么在下一个SYN时，包数量的增量inc为：inc = max, 1/1500) × 1500 / MSS，B为可用带宽的估算值（bits/秒），MSS为最大分段范围（bytes），SYN为 0.01秒。包的发送周期 P将被重新计算，计算公式为SYN / P = SYN / P’ + inc，P’为当前包的发送周期。

表1是一个UDT增量参数计算列表例子:

表1 UDT增量参数计算例子

B (Mb/s)	inc (packets)
B ≤ 0.1	0.00067
0.1 < B ≤ 1	0.001
1 < B ≤ 10	0.01
10 < B ≤ 100	0.1
100 < B ≤ 1000	1

根据列表的值域，并代入inc公式中计算，若 UDT想在一个丢失后恢复到 90%带宽的利用率，那么耗时将会是 7.5秒:[(L - 10n-1)/ (10n-9 * 1500 * 8) +(0.9 * L - (L - 10n-1)) / (10n-10 * 1500 * 8)] * 0.01= (0.9 * 10n) / (10n-9 * 1500 * 8) * 0.01= 750 SYN = 7.5s。相比之下，在 RTT为 200ms时，TCP需要 28分钟恢复到 1Gbit/s，4小时 43分钟恢复到 10Gb/s，以此看来，UDT的优势十分明显。

(4) 带宽估算

有效地利用网络带宽需要知道网络连接的确切信息，UDT采用以下带宽估算算法获取连接信息：每N个数据包（默认为 N=16），发送端不会等待下一个包的发送，相反地会发送两个连续的包（可称之为包对），且包与包之间没有任何延时和间隙。等到接收端收取这一包对的时候，再计算网络连接容量，并将其包含在下一个ACK中反馈回去。若 L为连接容量极值，C为当前传送速度，d为递减系数 1/9（NAK出现的比率），对于所有的流来说，当现在的传送率大于上一次减幅发生时的传送率时，带宽估算值是 L - C。在每一次减幅发生之后，传送率恢复完成之前，带宽估算值的将会在min{ L*d, L - C } 间波动。其中 L*d 代表当拥塞发生或事件丢失时，所有的流将对应的递减系数。由于 L值有可能小于 C值，所以在范围取值时要取可行的正数。

将带宽估算结合到瓶颈问题上，面对单个瓶颈问题(所有流都有将近一致的 L值)时，拥有较高传送率的流的速率无法更快地增加，此时UDT对流是公正公平的，而且比标准TCP中的 AIMD要快速。而面对多个瓶颈问题，UDT亦至少可以达到最少一半的公平性。

4 结束语

网络带宽的快速增加给新型传输协议的设计带来了强有力的挑战，既要求传输协议性能优越并易于使用、部署，又要求它们和已有传输协议保持兼容、友好，以达到更理想的使用效果。UDT很好的满足了这两方面的要求，UDT非常适合数据源少但数据密集型的应用，从而充分地利用网络带宽。UDT是开源软件，相信将来会有更多的网络应用采用UDT协议实现数据传输，同时，当多个UDT流共享同一网络链路时，UDT协议如何更精确地估计网络带宽是UDT拥塞控制算法的发展和改进重点。

参考文献

1 L. Xu, K. Harfoush, and I. Rhee. Binary Increase Congestion Control for Fast Long-Distance Networks. IEEE Infocom ‘04, Hongkong, China, Mar. 2004.

2 D. J. Leith and R. Shorten. H-TCP Protocol for High-Speed Long Distance Networks. PFLDnet ‘04, Feb. 2004.

3 K. Kumazoe, Y. Hori, M. Tsuru, and Y. Oie. Transport Protocol for Fast Long Distance Networks: Comparison of Their Performances in JGN. SAINT ‘04, Tokyo, Japan, 26~30 Jan. 2004.

4 D. Clark, M. Lambert, and L. Zhang. NETBLT: A high throughput transport protocol. ACM SIGCOMM ‘87, Stowe, VT, Aug. 1987.

5 Yunhong Gu and Robert L. Grossman UDT: UDP-based Data Transfer for High-Speed Wide Area Networks, Computer Networks (Elsevier). Volume 51, Issue 7. May 2007.

6 Yunhong Gu and Robert L. Grossman, Supporting Configurable Congestion Control in Data Transport Services, SC 2005, Nov 12 - 18, Seattle, WA, USA.

7 Yunhong Gu, Xinwei Hong and Robert L. Grossman, An Analysis of AIMD Algorithms with Decreasing Increases, First Workshop on Networks for Grid Applications (Gridnets 2004),Oct. 29, San Jose, CA, USA.

(收稿日期：2011-09-15)