UDT 协议在微震数据传输系统中的应用①

时美子[1] 冀常鹏[2](1.辽宁工程技术大学 研究生院 辽宁 葫芦岛 125105;
2. 辽宁工程技术大学 电子与信息工程学院 辽宁 葫芦岛 125105)

摘 要: 将 UDT 协议用于微震数据传输系统中，用于微震监测数据的实时可靠传输。通过对 UDT 协议的仿真及现场通信测试，验证了它在可靠性、实时性、吞吐量、CPU 占用率等方面的性能优势。

关键词: UDT; 微震监测; 微震数据传输; 拥塞控制; AIMD; DAIMD

Application of UDT to Microseism Data Transmission System

SHI Mei-Zi[1], JI Chang-Peng[2]
(1. Institute of Graduate, Liaoning Technical University, Huludao 125105, China;2 School of Electronic and Information Engineering, Liaoning Technical University, Huludao 125105, China)

Abstract: This paper applies the UDT to microseism data transmission system to make a real-time and reliable data transmission. By evaluating UDT's performance in both simulation environments and real networks, it has examined UDT's performance including reliability, real-time, throughput as well as the implementation efficiency.

Keywords: UDT; microseism monitoring; data transmission; congestion control; AIMD; DAIMD

1 引言

微震是矿岩破坏过程的伴生现象，其中包含了大量的有关围岩受力破坏以及地质缺陷活化过程的有用信息，故通过对微震信号的采集、处理、分析和研究，建立长期的微震事件数据库，通过对微震活动规律的统计与分析，预测岩土结构是否发生破坏，对防止顶板坍塌等事故的发生有积极意义。通过在地下及地面的岩土工程中布置传感器阵列，可以实现微震数据的自动采集、传输和处理，并利用定位原理确定破坏发生的位置，且在三维空间上显示出来。因此，微震监测技术具有远距离、动态、三维、实时监测的特点[1,2]。

在 2002~2008 年期间，我们在京煤集团的木城涧煤矿及大安山煤矿布置了 2 套自主研发的微震定位监测系统(共 12 个监测子节点)，用于对其矿区范围内的微震事件的监测、分析和研究。设备长期运行以来获取了大量宝贵的监测数据。微震监控中心通过对各监测点实时传回的微震数据进行综合分析，解算震源位置、震级等信息，并对其进行实时播报与入库保存。为煤矿的安全生产提供了有力的技术保障。

在整个系统中，对于微震数据的实时可靠传输的研究与实现部署直接关系到监测控制中心对微震监测数据的解算的准确性。进而，最终会影响到震源定位的精度。

在实时可靠传输的方案选择上，以往部署的系统中尝试了以下两种微震数据传输方案：

1) 基于 UDP 协议的实时微震数据传输方案；

2) 基于 TCP 协议的实时微震数据传输方案；

由于当微震事件到来时，微震数据传输系统瞬间传输的数据量会很大(通常为几十兆到几百兆不等)。因此，根据链路瞬时数据量变化大，需要的传输速率高等特点，我们在微震数据传输系统的设计初期采用了

① 基金项目:国家自然科学基金(50490275)
收稿时间:2009-06-30

基于 UDP 协议的设计方案。该方案具有高的数据传输速率，极低的 CPU 占用率等优点。但是，UDP 协议并不保证数据通信的可靠性。因此，在链路紧张的情况下，这套方案的传输误码率较高。

为了增加数据传输的可靠性，我们设计基于 TCP 协议的面向连接的微震数据传输方案。由于 TCP 协议加入了 AIMD 拥塞控制算法及出错重传机制，传输系统在数据的可靠性方面的到了大大的提高，系统在对设备及网络状况的实时监测能力也得到了提升。但此方案增加了数据采集端及监测中心服务器端的 CPU 的负荷，AIMD 拥塞控制算法在瞬时发送速率的提升上也不是很理想。

基于上述原因，我们在新的微震传输系统的设计上采用了基于 UDT(UDP-based Data Transfer Protocol)协议的支持。UDT 协议是一种基于应用层的面向连接的点对点单播的可靠传输协议。它集 UDP 与 TCP 各自优势，立足于效率、公平与稳定性。UDT 提供了与 TCP 类似的 Socket API，为程序的移植提供了极大的方便[3]。

2　UDT 协议

UDT 协议在网络层次结构中的位置完全位于用户空间即其位于 UDP 协议之上。它利用 UDP 协议传输用户数据和协议控制信息。利用分组测序算法来检验数据是否确实，以保证数据的可靠性。UDT 的主要目标是效率、公平、稳定。为了达到这些目标，UDT 进行了全新的架构与拥塞控制算法的设计[4]。UDT 协议的软件结构如图 1 所示。

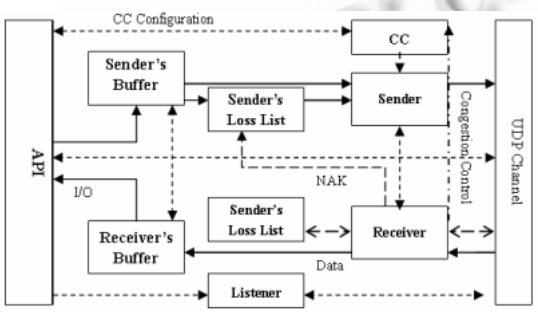

图 1　UDT 软件架构图

图 1 中给出了 UDT 实现的软件架构，除了发送端与接收端，软件架构中拥有一个与应用程序交互的 API 模块；一个基于 UDP 实现的数据包传输通道及收发端的数据缓冲模块。而且，为了在 UDP 的基础上增加可靠性与拥塞控制，在收发两端还各有一个模块用于管理数据丢失信息以及位于发送端的用于调整数据包发送速率的拥塞控制模块[5,6]。

3　UDT 协议的相关算法

3.1 UDT 协议的拥塞控制算法

当前使用最普遍的网络通信协议为 TCP 协议，其拥塞控制机制称为 AIMD 算法(加性增乘性减算法)。TCP 的性能与 RTT 有关，即长的 RTT 意味着 TCP 需要很长的时间才能够到达最大吞吐量。如果在这个期间发生了数据包丢失，那么 TCP 将永远也不能达到其最大吞吐量[7,8]。下面通过对 UDT 的拥塞控制算法推导来说明 UDT 的拥塞算法是如何解决上述问题的。

对于每一个速率控制间隔，如果没有收到接收端发来的响应(如丢包，延迟等)，则包的速率 x 在原有基础上增加 α(x)，对于任何响应，发送速率将会被常量因子 β 消减即：

$$x=x+\alpha(x),\ x=(1-\beta)\cdot x \tag{1}$$

其中，α(x)是单调递减函数，由 x 所引起的网络变化为：

$$x(t+1)-x(t)=p(0)\cdot a(x(t))-\sum_{i=1}^{x(t-D)}(p(i)\cdot(1-(1-\beta)^i))\cdot x(t) \tag{2}$$

式中，p(i)为(t, t+1)期间内发生 i 个丢包事件的概率；D 表示网络回环延迟；$p(i)\cdot(1-(1-\beta)^i)$表示当 i 个包丢失时，发送速率可能的减少量。另外，在稳定的状态下，x(t)与 x(t-D)的差异非常小，设 x(t)= x(t- D)，则公式(2)被化简为：

$$x=(1-x(t)\cdot p(t))\cdot\alpha(x(t))-x(t)\cdot p(t)\cdot\beta\cdot x(t) \tag{3}$$

其差分函数为：

$$x=k(x)\cdot(U'(x)-p(t)) \tag{4}$$

其中 $k(x)=x^*\alpha(x)+\beta^*x^2$ 对于任何一个 x (x> 0)是一个恒正且单调递增函数。

$$U(x)=\int\frac{\alpha(x)}{x\cdot\alpha(x)+\beta\cdot x^2}dx \tag{5}$$

公式(5)称为 DAIMD 算法的效用函数。且 U〃(x)<0，根据 Srikant 的理论，公式(4)这个拥塞控制算法最终将达到一个稳定的状态。UDT 的速率控制算法是 DAIMD 算法的一种特殊形式，即指定 α(x)为：

$$\alpha(x)=10^{\lceil\log(L-c(x))\rceil-\tau}\times\frac{1500}{S}\times\frac{1}{SYN} \tag{6}$$

公式(6)中 τ 为协议参数，UDT 协议中指定其为 9。UDT 中规定 1500 比特为数据包的标准大小。公式(6)的曲线图如图 2 所示。

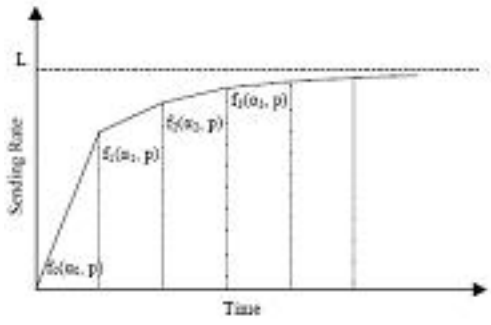

图 2 发送速率随时间变化曲线

由图 2 可以看出，在开始阶段，UDT 协议使发送速率快速增加。随后，发送速率增量渐渐慢下来直至发送速率的大小接近链路的连接能力。这一过程所需的时间要远远小于基于TCP协议的实现完成相同的过程的时间。

为了进一步验证 UDT 协议的效率特性，假设在 k 段(k=0,1,2,3,…)，f(k)为数据的吞吐量函数，递增参数为 α(k)，丢包率为 p，e 满足 $10^{e-1}<L\leqslant 10^{e}$。

根据速率差分函数(公式(4))得，UDT 在任意阶段 k(xk*)的稳定方程为：

$$x_k^* = \frac{1}{2\beta}\left(-\alpha_k + \sqrt{a_k^2 + \frac{4\beta a_k}{p}}\right) \approx \sqrt{\frac{\alpha_k}{\beta \cdot p}} \tag{7}$$

由于 $\alpha_0=\alpha(0)$，$\tau=9$，$\beta=1/9$，所以上式可化简为

$$x_k^* = \frac{3}{\sqrt{p}} \cdot 10^{\frac{-k+e-9}{2}} \tag{8}$$

这里 x_k 的单位是 Packets / SYN (SYN 是同步时间间隔)，其中 x_k 是吞吐量的函数，它的单位是 bits / sec。则：

$$x_k^* = \frac{1}{SYN} \cdot \frac{1500}{S} \cdot x_k^* \cdot S \cdot 8 = \frac{3.6}{SYN} \cdot \frac{1}{\sqrt{P}} \cdot 10^{\frac{e-k+1}{2}} \tag{9}$$

从上式可以看出当发送速率增至某一值时，UDT 的拥塞控制会进入下一阶段，但是当 k 增大时，处于稳定状态的吞吐量 x_k^* 将随之减少。当 $L-\frac{3.6}{SYN}\cdot\frac{1}{\sqrt{P}}\cdot 10^{\frac{e-k+1}{2}} \geq 10^{e-k-1}$ 时，k 会停止增加。满足上述条件的最小 k 值则是 UDT 数据流的一个稳定阶段。此外，当 p=0 时，公式(2)变为：

$$x_k(t+1) = x_k(t) + a_k \tag{10}$$

这个线性的增长告诉我们发送速率增长的每个阶段都需要一个固定的时间间隔来使下一阶段继续增长。特别是 UDT 流从 0%增至链路连接能力的 90%这一个阶段。若 UDT 在第一阶段的最后达到 R_0($R_0\leqslant 0.9L$)，那么 R_0 达到 0.9L 花费的时间为：

$$\left(\frac{R_0}{a_0} + \frac{0.9L - R_0}{a_1}\right) \cdot \frac{SYN}{1500\times 8} = \frac{9\cdot(L-R_0)}{a_0} \cdot \frac{SYN}{1500\times 8} \tag{11}$$

由于 $L-R_0=10^{e-1}$，$a_0=10^{e-9}$，SYN=0.01，所以公式(11)的最终结果为 750SYN，即 7.5s。而 TCP 协议在 RTT 同样为 200ms 的情况下，发送速率要达到 1Gbit/s，理论上则要花费 28min。可见 UDT 在其效率方面要远远优于 TCP 协议[4]。

3.2 UDT 协议的丢包处理算法

通常，网络丢包现象被认为是一种简单的辨别网络拥塞的方法。常见的引起网络丢包的原因有：

1) 由于网络失去同步引起丢包。

2) 由于链接错误引起的丢包。

UDT 协议在解决上述这两种丢包情况时应用的算法如表 1 所示，其中 LSD 为接受到的最后一个 NAK 时，包的最大序号；STP 为包的发送周期；AvgNAK 用于记录当前拥塞的 NAK 个数；NumNAK 则是其滑动平均值，DR 是[1，AvgNAK]间的随机数。

表 1 UDT 的丢包处理算法

1．如果 NAK 中的最大丢失序号大于 LSD：

A) 将 STP 增加 1/8，即 STP=STP×(1+1/8)；

b) 更新 AvgNAK=(AvgNAK×7+NumNAK)/8；

c) 更新 DR，即 DR=rand(AvgNAK)；

d) 重置 NumNAK，即令 NumNAK=0；

e) 记录 LSD；

2．否则，将 NumNAK 增加 1，并检测 NumNAK%DR 是否为 0，若为 0，则：

a) 将 STP 增加 1/8，即 STP=STP×(1+1/8)；

b) 记录 LSD；

UDT 的 NAK 机制不同于 TCP 的重复确认 ACK 机制。所有的丢包事件都会被 NAK 记录下来[4]。

4 仿真实验

实验仿真平台的搭建我们采用 Ubuntu 8.10 操

作系统以及开源免费的 NS-2 网络仿真软件。传输系统的实验拓扑图如图 3 所示。该拓扑图中 0 为 TCP 发送节点，1 为 UDT 发送节点[9]。分别对带宽与连接错误率、吞吐量变化率进行了性能仿真。

图 3　UDT 实验拓扑图

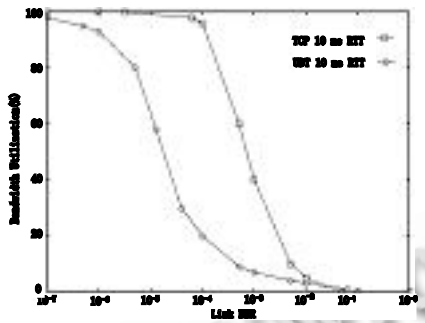

图 4　带宽利用率与连接错误的关系

从图 4 中我们可以看出，UDT 在带宽利用率达到 80%时，其连接错误率仅仅为 10^{-5}，而 TCP 在相同的带宽利用率下却仅仅达到了 10^{-3}。这说明了 UDT 的丢包处理算法使带宽利用率在高连接错误率条件下得到了很大的改善。

图 5 给出了 UDT 协议与 TCP 协议吞吐量性能上的表现。从图 5 中可以清楚地看到 UDT 协议达到 900Mbps 的时间仅为 6s 左右(RTT=100ms)，而在相同条件下 TCP 达到相同的吞吐量却用了 27s。可见 UDT 协议在吞吐量的性能方面要远远优于 TCP 协议，尤其是在高速数据传输的情况下。

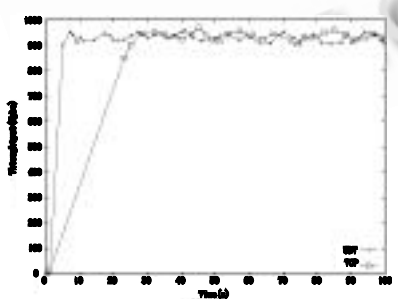

图 5　吞吐量随时间变化关系

5　现场测试

在现场测试中，我们在位于 820 井口微震监测点(与监测中心相距约 4000m，RTT 约为 100ms)进行了数据传输实验，该监测点的单次监测数据大小为 120M，以下实验在4种不同的传输速率下各完成200次传输实验。表 2 中给出了 4 种传输速率下的误码率平均值。从表 2 中可以看出，在传输速率为 100kbps 时，UDP 性能最好。但随着传输速度的增加 UDT 的可靠性逐渐提升，近似于 TCP 的可靠性水平，而没有拥塞控制的 UDP 此时已无法保证数据的可靠传输了。表 2 表明 UDT 的拥塞算法发挥了作用，使传输系统在可靠性能上得到了很大提升。

表 2　传输速率与误码率的关系

通信协议 / 传输速率	UDP	UDT	TCP
100Kbps	0.8×10^{-12}	1.1×10^{-12}	1.0×10^{-12}
5Mbps	2.3×10^{-10}	1.3×10^{-12}	1.6×10^{-12}
10Mbps	5.7×10^{-9}	2.4×10^{-12}	2.2×10^{-12}
30Mbps	7.9×10^{-9}	3.9×10^{-11}	3.7×10^{-11}

另外，我们对 UDP、UDT、TCP 协议在传输速率为 10Mbps 情况下的 CPU 占有率也进行了测试，如图 6 所示。从图 6 中可以看出，UDT 协议的拥塞控制算法使其对 CPU 的占有率明显低于 TCP 协议而略高于无拥塞控制的 UDP 的协议。

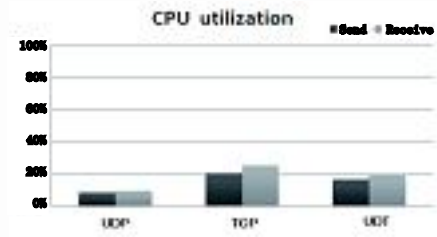

图 6　三种协议的 CPU 利用率

6　结论

本文首次将 UDT 协议用于微震数据传输系统中，通过对 UDT 协议拥塞控制算法的推导，从理论上证明了其在效率，公平，可靠性上的优势。并利用 NS-2 网络仿真软件对其非拥塞错误下的带宽利用率及吞吐量的性能进行了仿真。通过现场测试，对其可靠性、实时性及 CPU 利用率进行了性能评价。实验结果充分的说明了 UDT 协议优越性。UDT 协议虽然优点众多，但作为一个新的实时可靠通信协议，其成熟性与稳定性还有待于今后的长期测试。

参考文献

1 黄维新,王李管,唐礼忠,等.基于微震监测技术在矿山

(下转第 216 页)

(上接第 178 页)

安全管理中的应用研究.中国安全科学学报, 2008,18 (1):165－170.

2 潘一山,赵扬锋,官福海,李国臻,马植胜.矿震监测定位系统的研究及应用.岩石力学与工程学报, 2007, 26(5):1002－1012.

3 Gu YH, Robert L. Grossman. Supporting Configurable Congestion Control in Data Transport Services. SC05, Nov 12–18, 2005, Seattle, WA, USA.

4 Gu YH, Robert L. Grossman: Optimizing UDP-Based Protocol Implementations. Proc. of the Third International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet2005), Feb 3-4, 2005, Lyon, France.

5 Gu YH. A High Performance Data Transport Protocol. [Ph.D. Thesis]. The Graduate College of the University of Illinois, Chicago, 2005.

6 Gu YH, Hong XW, Grossman R. Experiences in Design and Implementation of a High Performance Transport Protocol. SC04, Nov 6-12, Pittsburgh, PA, USA.

7 Gu YH, Hong XW, Grossman R. An Analysis of AIMD Algorithms with Decreasing Increases. Gridnets 2004, First Workshop on Networks for Grid Applica- tions, Oct. 29, San Jose, CA, USA.

8 LOWSH, PAGANINIF, DOYLEJC. Internet congestion control. IEEE Control Systems Magazine, 2002, 32 (1):28－43.

9 于斌,孙斌,温暖,王绘丽,陈江锋. NS2 与网络模拟.北京:人民邮电出版社, 2007.