文章编号: 1001-0920(2011)04-0513-06

基于数据流的Internet网络控制系统延时模型研究

程 论, 王中杰

(同济大学 控制理论与控制工程系，上海 200092)

摘 要: 在给出数据流定义的基础上, 从理论上分别针对传输控制协议(TCP)和用户数据包协议(UDP)提出了基于数据流的Internet网络控制系统(INCS)延时模型. 数据流的定义描述了INCS中所有数据流的输入和输出特性, 网络延时模型描述了TCP和UDP协议的传输延时特性. 此外, 提出了基于Transit-Stub的INCS延时测量模型, 并通过实验验证了基于数据流的INCS延时模型的正确性. 分析了INCS中不同类型的数据流适用的传输协议, 从而为设计INCS提供了理论依据.

关键词: 网络控制系统；数据流；延时模型

中图分类号: TP273 **文献标识码:** A

Research on time-delay model for NCS based on data streams

CHENG Lun, *WANG Zhong-jie*

(Department of Control Theory and Control Engineering，Tongji University，Shanghai 200092，China. Correspondent: CHENG Lun，E-mail：yuncll@163.com)

Abstract: The definition of data stream is given, based on which the time-delay model for Internet network control system(INCS) is proposed theoretically for both TCP and UDP protocol. The definition of data stream describs the input characteristics and output charactersitics for all the data streams. The time-delay model describes the time-delay characteristics for TCP and UDP protocol. The time-delay measurement model for INCS is proposed based on Transit-Stub. Experiment results show the validity of time-delay model. As a result, theoretical basis is provided for designing an INCS.

Key words: networked control systems；data stream；time delay model

1 引 言

自网络控制系统(NCS)提出以来, 其延时问题便成为NCS走向实用化的重大障碍. 由于局域网的延时基本上可以确定, 目前NCS延时主要指基于Internet的网络控制系统(INCS)的延时. Internet的优点是跨越地域性广和融合许多异构网络. 但正是这一特点使得Internet延时受到许多因素的影响, 导致确定Internet延时比较困难, 而确定延时几乎是所有控制系统设计和研究的关键, 所以对INCS的延时研究显得尤为重要.

现有INCS延时的研究比较少, 虽然在Internet协议方面已有一些文献, 但主要是针对传输协议进行独立分析和研究[1-2], 没有考虑INCS交互数据的特性和要求. 另外, 在提高控制系统的传输性能方面有一些研究, 但采用的技术和方法不适用于Internet环境. 文献[3]在讨论媒体访问控制(MAC)层的数据传输技术时, 将数据流分为周期性数据、猝发性数据以及非实时性数据, 并讨论了各种MAC层传输技术用于传输3种数据流的优缺点. 但采用该传输技术实现的INCS无法在Internet中提高性能. 文献[4]通过仿真实现了基于用户数据包协议(UDP)传输周期性数据, 使用传输控制协议(TCP)传输非实时性数据, 循环组合使用TCP和UDP传输猝发性数据的过程. 但该文没有证明各种数据在不同协议下的传输特征, 更没有考察不同背景流量载荷下数据流传输延时的特征.

本文首先根据数据流的输入和输出特性, 提出一个统一的INCS数据流定义; 然后, 在理论上分别针对TCP协议和UDP协议提出了INCS的延时模型, 在该模型中, 描述了TCP和UDP协议的传输延时特性; 同时, 提出一种基于Transit-Stub的INCS延时测量模

收稿日期: 2010-01-04；**修回日期:** 2010-05-30.

基金项目: 教育部新世纪优秀人才支持计划项目(NCET-06-0382)；教育部重大项目(306023)；教育部博士点基金项目(20070247075).

作者简介: 程论(1983−), 男, 博士生, 从事网络控制系统、Internet测量等研究；王中杰(1971−), 女, 教授, 博师生导师, 从事复杂生产过程、网络控制等研究.

型, 并通过实验验证了上述理论延时模型的正确性; 最后, 分析了INCS中不同类型的数据流实用的传输协议, 从而为设计控制系统提供了依据.

2 INCS的数据流定义

通过对INCS结构的深入分析和多次INCS通讯协议的实际设计开发, 本文给出了INCS通用数据流(data Stream)定义, 以指标参数的形式抽象了各种数据流的输入特性和输出特性.

2.1 数据流定义

数据流是逻辑上相关的一系列数据, 按照一定规则或限制, 由源节点(Src)出发经过网络并到达目的节点(Dest), 数据流是一个单向的矢量.

定义 1 流Stream是六元组, 记为Stream $= \langle S, I_s, N_{lc}, N_l, T_l, T_0\rangle$, 其中:

1) $S = \{S(t)|\forall t \geqslant T_0, S(t)$ 为流中 t 时刻包长度的随机变量$\}$, 且 S 满足:

① $\{S(t), t \geqslant T_0\}$ 为一随机过程;

② $\forall T_0 \leqslant t_1 < t_2$, 随机变量 $S(t_1)$ 与 $S(t_2)$ 同分布.

2) 随机序列 $I_s = \{I_s(n), n \in Z|Is(n)$ 为流中包 P_n 与 P_{n+1} 的发送时间间隔 $\}$, 且 $\forall m, n \in Z^+$, 随机变量 $I_s(m)$ 与 $I_s(n)$ 独立同分布.

3) $T_0 \in R^+$ 为流中初始包的发出时间.

4) 随机序 $N_{lc} = \{N_{lc}(n), n \in Z^+|N_{lc}(n)$ 为流中第 n 次连续丢包次数的随机变量$\}$, 且 $\forall m, n \in Z^+$, 随机变量 $N_{lc}(m)$ 与 $N_{lc}(n)$ 独立同分布.

5) 时间段 $T = [T_1, T_2]$ 内的丢包次数 $N_l = \sum\limits_{t_i \in T} N_{lc}(i)$, t_i 为第 n 次丢包的时间.

6) $T_l \in R^+$ 为流中判断包丢失的时间阈值.

按照如上定义可知, 上述参数中与源节点相关的参数有 S, I_s 和 T_0; 与目的节点相关的参数有 N_{lc}, N_l ($[T_1, T_2]$ 间隔的丢包次数) 以及 T_l.

2.2 INCS数据流的分类

INCS各个组成部分之间的数据流可按不同的规则进行分类, 文献[4]将数据分为以下3种数据, 按照上述定义其形式化描述如下:

假设 $F(t)$ 是一个关于时间的随机分布函数, i, s, m, n, E, e 是大于零的常数, 且 $e < E$.

1) 间隔数据

I 服从 $I_s(i) = 1$ 分布; S 服从 $S(s) = 1$ 分布; N_{lc} 在区间 $[0, n]$ 内服从 $F(t)$ 分布, N_l 在区间 $[0, m]$ 内服从 $F(t)$ 分布, 其中 m 和 n 视具体系统而定; $T_l > 0$.

2) 紧急数据

I, S 服从 $F(t)$ 分布; N_{lc} 服从 $N_{lc}(0) = 1$ 分布; N_l 服从 $N_l(0) = 1$ 分布; $0 < T_l < E$.

3) 可靠数据

I, S 服从 $F(t)$ 分布; N_{lc} 服从 $N_{lc}(0) = 1$ 分布; N_l 服从 $N_l(0) = 1$ 分布; $0 < T_l < e$.

本文根据对INCS开发的经验可知, 将INCS中的数据流按照逻辑相关性分类更加合理, 可归为如下5类: 采样流、用户控制流、算法控制流、紧急信息流和系统诊断流. 根据数据流的定义可形式化描述如下:

1) 采样流

传感器按照指定间隔发送数据至控制器. 特点是数据发送周期固定, 每次数据量相对较小, 数据具有时效性. 形式化描述为: I 服从 $I_s(i) = 1$ 分布; S 服从 $S(s) = 1$ 分布; N_{lc} 在 $[0, n]$ 之间, N_l 在 $[0, m]$ 之间服从 $F(t)$ 分布, 其中 m, n 由具体系统确定; $T_l > 0$ 由具体系统确定.

2) 用户控制信号

用户根据需要对控制器发送命令信号, 控制器作出判断并向执行器或传感器发送命令, 强调数据的可靠性,实时性要求不高. 形式化描述为: I, S 服从 $F(t)$ 分布; N_{lc} 服从 $N_{lc}(0) = 1$ 分布; N_l 服从 $N_l(0) = 1$ 分布; $0 < T_l < E$.

3) 算法调整信号

系统在动态过程中, 控制器按照控制算法向执行器发送调整命令. 数据的发送频率与具体算法相关, 一般表现无规律. 执行器在接收该数据时具有时效性. 形式化描述为: I 服从 $F(t)$ 分布; S 服从 $S(s) = 1$ 分布; N_{lc} 在 $[0, n]$ 之间, N_l 在 $[0, m]$ 之间服从 $F(t)$ 分布; $T_l > 0$ 由具体系统确定.

4) 紧急操作信号

在紧急情况下, 控制器快速发送至执行器或报警设备. 该数据实时性要求较高, 而且需要快速、可靠地发送至目标. 形式化描述为: I, S 服从 $F(t)$ 分布; N_{lc} 服从 $N_{lc}(0) = 1$ 分布; N_l 服从 $N_l(0) = 1$ 分布; $0 < T_l < e$.

5) 系统诊断数据

控制器定时或不定时发送一些诊断数据至传感器和执行器; 分析返回数据, 以了解执行器和传感器的运行和故障状态. 该数据的周期一般比采样数据长, 数据量比其他类型数据大, 数据的逻辑完整性和可靠性要求较高, 但响应时间相对较长. 形式化描述为: I 服从 $I_s(i) = 1$ 分布, S 服从 $S(s) = 1$ 的分布, N_{lc} 服从 $N_{lc}(0) = 1$ 分布, N_l 服从 $N_l(0) = 1$ 分布, $0 < T_l < E$.

上述描述定义了INCS的数据流, 形式化地给出

了 INCS 的网络输入数据和系统中不同数据的输出要求. 对网络传输协议建模后, 结合数据流的形式化定义, 可将其应用于 INCS 性能的理论分析和评估. 通过理论推导和证明可知, INCS 系统在具体环境下应选用何种传输网络和协议更有利于性能的提升.

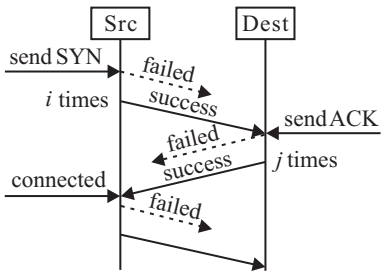
3　TCP/UDP 延时模型

Internet 中几乎所有应用都是基于 TCP 和 UDP 协议传输数据. INCS 的交互数据与其他 Internet 应用相比完全不同, 主要区别表现在: INCS 对数据传输的实时性要求较高, 而且每次传输的数据包较小, 因此在选择 INCS 的数据传输协议时, 必须根据协议的特性与数据流的性质进行详细分析来确定.

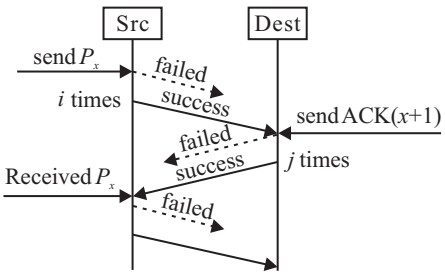
3.1　TCP 延时模型

TCP 支持大数据和小数据传输, 对两种数据使用不同的传输策略, 同时还提供了许多套接字选项和优化算法[5]. 由于 INCS 的数据包较小, 一般仅为几个到几十个字节, 通常选用小数据传输模式; 为了提高数据传输的实时性, 使用 TCP 的无延时选项且需要禁止尼尔算法.

每个 TCP 在发送数据前, 需要 3 次握手: 源节点发送 SYN 到目的节点, 然后等待, 直到收到目的节点发送的确认 (ACK) 信号, 才可认为连接已经建立 (如图 1(a) 所示).



(a) TCP连接过程



(b) TCP数据包传输过程

图 1　TCP 的连接和数据传输

数据从 S 源节点成功传输至目的节点的过程描述如下: 源节点发送序列号 n 的数据包至目的节点, 然后源节点开始等待接收并确认收到信号, 同时, 源节点已有一个定时器定时查询源节点是否收到定时节点对序列号 n 数据包的确认信号. 如果定时器超时, 则源节点重传一次数据包, 并两倍延长定时器的超时时间, 直到增加到最大超时为止. 如果收到 ACK 信号, 则认为数据传输成功.

如果目的节点收到源节点发送的数据包, 则立刻发送一个序列号为 $n+1$ 的确认 (ACK) 信号, 然后开始等待下一次数据包的发送过程. 同上, 目的节点也根据相同规则重传 ACK 信号. TCP 的数据包传输过程如图 1(b) 所示.

根据以上分析, TCP 的传输延时可建模如下:

假设 p_s 为源端到目的端单向丢包概率, p_d 为目的端到源端单向丢包概率. 源端传输数据成功的概率为 $p_h(i,j)$. 其中: i 为数据从源端到目的端传输失败的次数, j 为确认信号 ACK 从目的端到源端传输失败的次数. RTT 为双向延时, T_s 为 TCP 发送超时时间的基数. 则有

$$p_h(i,j)=p_s^i(1-p_s)p_d^j(1-p_d). \tag{1}$$

传输过程的延时为

$$D_h(i,j)=\mathrm{RTT}+\sum_{k=0}^{i-1}2^kT_s+\sum_{k=0}^{j-1}2^kT_s=\mathrm{RTT}+(2^i+2^j-2)T_s. \tag{2}$$

在时间 t 内, 传输过程的成功概率为

$$p\{D_h(i,j)\leqslant t\}=\sum_{D_h(i,j)\leqslant t}p_h(i,j). \tag{3}$$

根据式 (1) 和 (2), 推导出该传输过程的平均延时为 [4]

$$E(D_h(i,j))=\sum_{D_h(i,j)\leqslant t}(p_h(i,j)\cdot D_h(i,j))\approx \mathrm{RTT}+T_s\Big(\frac{1-p_s}{1-2p_s}+\frac{1-p_d}{1-2p_d}+2\Big),\quad 0<P_s<1/2, 0<P_d<1/2. \tag{4}$$

3.2　UDP 延时模型

UDP 的数据传输过程比 TCP 相对简单, UDP 是无连接的, 数据发送过程只有一种方式, 即数据从源端发送至目的端. UDP 是不可靠的, 源端不需要接收目的端收到数据包的确认信号, 也没有重传机制. UDP 是无拥塞控制的, UDP 不关心源端和目的端的缓冲大小以及网络中正在发送的数据量大小.

UDP 数据的成功传输过程如图 2 所示, 源端发送数据包至目的端, 前 i 次发送失败, 第 $i+1$ 次成功, 其中目的端确定数据是否丢失是由目的端接收定时器的超时所决定.

UDP 的延时模型建模如下. 假设 p_s 为源端到目的端单向丢包概率; 源端传输数据成功的概率为 $p_h(i)$, 其中 i 为数据从 Src 到 Dest 传输失败的次数;

OWD 为单向延时. 则有

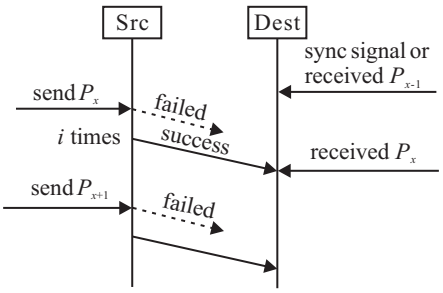


图 2 UDP 数据包的传输过程

$$p_h(i) = p_s^i(1-p_s). \tag{5}$$

源端到目的端数据成功传输过程的时延为

$$D_h(i) = (\mathrm{OWD} + iT_s), \tag{6}$$

$D_h(i)$ 在 t 内成功发送的概率为

$$p\{D_h(i) \leqslant t\} = \sum_{D_h(i)\leqslant t} p_h(i). \tag{7}$$

由式 (4) 和 (5) 可推导 UDP 传输的平均延时为

$$E(D_h(i)) = \sum_{D_h(i)\leqslant t} (p_h(i)\cdot D_h(i)) =$$

$$\mathrm{OWD} + \frac{p_s}{1-p_s}T_s = \frac{\mathrm{RTT}}{2} + \frac{p_s}{1-p_s}T_s. \tag{8}$$

上述两个延时模型 (4) 和 (8) 必须在统一的假设和约束条件下, 才可比较数据流使用两种传输协议传输数据的平均延时, 即:

1) 假设 INCS 系统是稳定的或 INCS 系统的可靠概率为 R;

2) 数据流 S 在 Internet 中传输, 即 S 是网络的输入;

3) 数据包的双向传输路径相同, 即 $\mathrm{OWD} = \mathrm{RTT}/2$;

4) 数据包在双向路径中的丢包概率相等.

4 基于 Transit-Stub 结构的 INCS 延时测量模型

合理的 Internet 建模对 Internet 技术的开发和性能分析非常重要, 而完整仿真 Internet 则是不可能的. 本文针对 INCS 的 Internet 应用, 构建了合适的延时测量模型, 主要包括: 主路径 Transit-Stub 拓扑结构、网络基本参数和数据流的发送规律.

4.1 拓扑结构和网络相关参数的设定

现有对 Internet 拓扑结构的研究和仿真大都是以研究路由算法为目的, 使用较为广泛的是文献 [6] 为比较 Minimum Steiner Tree 算法而提出的模型, 随后文献 [7] 又提出了更好的 Transit-Stub 模型. 本文根据 Transit-Stub 模型的结构特点, 将 Transit-Stub 模型简化抽象为以一条主要路径为主, 多条短路径为辅的主路径 Transit-Stub 拓扑结构 (如图 3 所示). 主路径是 $P_{(1,n)}$, 其中 n 为主路径的节点数; 源节点和目的节点是 N_1 和 N_n. 还有 $n-3$ 条辅助路径 $P_{(i,i+n-3)}$, 源节点和目的节点是 N_i 和 $N_{(i+n-3)}$, 其中 $n < i < n-2$.

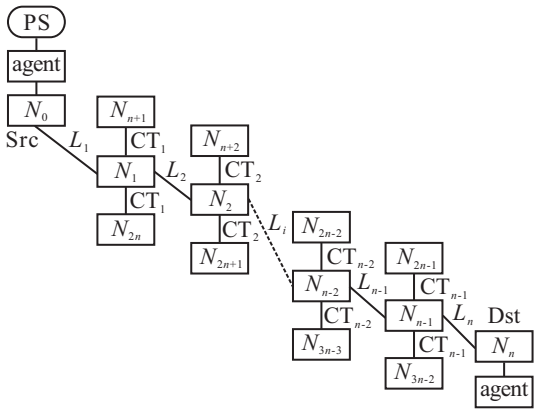


图 3 基于 Transit-Stub 的延时模型结构

路径 $P_{(1,n)}$ 中每个连接 $L_{(1,n-1)}$ 的带宽 B 均符合如下规律:

$$B_1 > B_2 < \cdots \leqslant B_i \leqslant \cdots < B_{(n-1)/2} >$$

$$\cdots \geqslant B_j > \cdots > B_{(n-2)} < B_{n-1},$$

$$2 < i < \frac{n-1}{2},\ \frac{n-1}{2} < j < n-2. \tag{9}$$

固定传输延时必须适当考虑链接长度带来的延时影响. 具体规律如下:

$$T_1 \leqslant T_2 \geqslant \cdots \geqslant T_i \geqslant T_{(n-1)/2} <$$

$$\cdots \leqslant T_j \leqslant \cdots \leqslant T_{n-2} \geqslant T_{n-1},$$

$$2 < i < \frac{n-1}{2},\ \frac{n-1}{2} < j < n-2. \tag{10}$$

基于 Transit-Stub 模型和网络实际参数, 式 (9) 和 (10) 定义了每个链接的固定传输延时和带宽的设定, 为后续的网络数据流构建提供了网络环境基础.

4.2 模型中的数据流

网络的拓扑结构是延时测量模型的基础, 其形式将影响网络数据流的模拟; 同时网络的带宽和固定延时与传输数据流的延时密切相关, 构建合适的数据流对本模型是非常重要的. 网络数据流包括两部分: 探测流和背景流. 探测流运行在主路径 $P_{(1,n)}$ 中, TCP 和 UDP 两种类型的背景流则运行在所有路径中.

为分析 TCP 和 UDP 在不同网络负载下的传输延迟, 本文定义了网络空闲和网络繁忙两种状况, 即

$$R_i + R_{pt} \leqslant \mathrm{Min}(B_i),\ 1 < i < n-1; \tag{11}$$

$$R_i + R_{pt} > \mathrm{Min}(B_i),\ 1 < i < n-1; \tag{12}$$

$$R_i = \mathrm{UDP}(R_i) + \mathrm{TCP}(R_i).$$

其中: R_i 为链接 L_i 的背景流速率, 背景流包含 UDP 和 TCP 两种类型; R_{pt} 为 $P_{(1,n)}$ 中的探测流速率. 式 (11) 为网络空闲状态, 表示主路径中所有 Link 中的背

景流速率与探测流速率之和小于等于对应链接的带宽; 式 (12) 为网络繁忙状态, 表示主路径中任何一个连接的背景流速率与探测流速率之和大于对应链接的带宽.

文献 [8] 统计了 24 h 内 Internet 主干道中流量信息, 其中: 流量分布的 90% 是 TCP, 其他是 UDP. 本模型的 Backbone 连接 ($L_{(n-1)/2}$) 的流量采用文献 [8] 的结果. 模型中其他路径中背景流的 TCP 与 UDP 的比例采用帕累托最优原则[9], 即 80% 为 TCP 数据包, 其他为 UDP 数据包.

5　TCP/UDP 延时分析

基于第 3 节和第 4 节的模型构造实验环境, 分别计算 TCP 和 UDP 的传输延时和丢包率, 从而验证了第 3 节中的理论建模结果, 即 TCP 的平均延时大于 UDP 的平均延时; 丢包率越大的网络 (繁忙的网络环境), TCP 平均延时比 UDP 平均延时更大.

5.1　实验设计

实验环境是 x86 平台下的 Linux, 内核版本为 2.6.18, NS-2[10] 版本为 2.33, Gawk[11] 为 3.15. 分别进行两组实验: 一组是在网络空闲状态下, 另一组是在网络高负载下. 根据 INCS 的数据流特点, 实验中 UDP 探测流的速率为 100 Kbps, 数据包大小为 50 B, 间隔为 4 ms; TCP 探测流的速率为 100 Kbps, 数据包大小为 1 024 B, 间隔为 80 ms. 因为越靠近网络主干道的链接 Link, 其数据流量越稳定, 所以本实验背景流主要使用固定速率数据流 CBR, 大小在正负 10% 的范围内变化. 但第 2 个链接 (L_2) 和倒数第 2 个链接 (L_{n-2}) 使用指数分布发送流. 实验的详细参数见表 1.

表 1　仿真实验参数信息

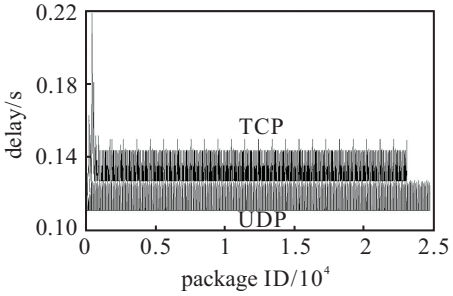
探测流类型	TCP	UDP
速率/Kbps	100	
包大小/B	1024	50
主路径的节点数	10	
链接的带宽/bps	$L_1 = L_9 = 100$M, $L_2 = L_8 = 1$M, $L_3 = L_4 = L_6 = L_7 = 300$M, $L_5 = 1$G	
链接的固有延时/ms	$T_1 = T_9 = 5, T_2 = T_8 = 20,$ $T_3 = T_4 = T_6 = T_7 = 10,\ T_5 = 20$	
探测流接收对象	NULL	TCPSINK
背景流类型	TCP UDP	

从 NS-2 的 trace 文件中获取 UDP 结果信息比较简单; 而获取 TCP 数据包的延时和丢包率则较为复杂, 必须提取 ACK 的发送、丢失和接收次数, 反推 TCP 发送数据包, 待确定 TCP 数据的正常发送后, 再计算延时和丢包率. 实验结果见表 2.

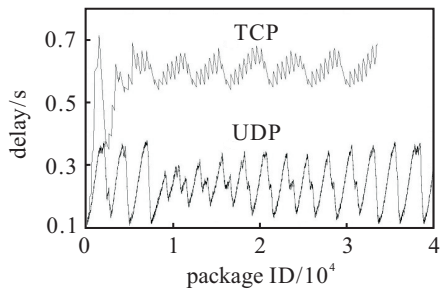
表 2　实验结果

网络状态	空闲		繁忙	
探测流类型	TCP	UDP	TCP	UDP
实验持续时间/s	50	50	50	50
发送数据包数	3 956	11 001	401	11 001
丢包数	0	0	31	425
接收数据包数	3 945	11 001	363	10 576
ACK的发送数	3 945	–	363	–
ACK的丢失数	0	–	0	–
接收到ACK数目	3 936	–	363	–

图 4 显示了多次实验后的 TCP 和 UDP 传输数据的平均延时信息. 其中: 横坐标是包的序号; 纵坐标是延时大小.



(a)　网络空闲



(b)　网络繁忙

图 4　空闲和繁忙网络下的 TCP/UDP 平均延时比较

5.2　实验结果分析

两种网络负载状态下的多次实验结果表明, UDP 数据包的平均单向延时小于 TCP 数据包. 空闲状态下 UDP 延时为 110 ~ 128 ms, TCP 在稳定状态下为 128 ~ 145 ms, 二者丢包率均为 0; 繁忙状态下 UDP 的单向延迟为 110 ~ 350 ms, 而 TCP 在稳定状态下为 550 ~ 680 ms, TCP 和 UDP 的丢包率都明显增加, 分别为 7.7% 和 3.8%.

值得注意的是, 由于 TCP 协议自身和实验环境的限制, 实际情况中 TCP 在繁忙网络下会发生丢包现象, 但是丢包率比仿真实验小. 具体原因如下:

1) TCP 是可靠的传输协议, 但在繁忙的网络情况下, 可能会发生多次重传失败, 进而发生丢包现象;

2) 本实验是在固定时间内的测量, 由于网络拥堵

使许多ACK没能在这段时间内发送至Src, 在统计过程中便认为其丢包;

3) NS-2对TCP协议仿真的限制, 也会导致丢包的增加.

在网络空闲情况下, TCP的平均单向延时大于UDP, 主要原因是在相同网络背景流量下, TCP的ACK机制造成了网络的额外负载.

在网络高负载情况下, TCP性能显然不如UDP. 虽然UDP包的平均单向延时相对于空闲状态下增加最大约2倍, 平均丢包率也有所增加, 但TCP包的平均单向延时增加最大约6倍于网络空闲. 主要原因是TCP的可靠保证机制引发了严重的性能缺陷. 因此, 在网络繁忙的状态下, TCP不适合传输数据包小且速率大的数据流.

6 结 论

本文首先提出了INCS中数据流的形式化定义, 并对INCS的数据流进行了详细分类, 在此基础上, 分别建立了TCP和UDP延时模型; 然后借鉴路由分析中的Transit-Stub结构, 提出了基于Transit-Stub结构的INCS延时测量模型; 最后, 通过实验表明了TCP和UDP两种协议在Internet环境下的传输性能, 并从理论和实验角度证明了INCS中不同数据传输协议的延时性能.

根据理论计算和实验结果可知, 对INCS不同类型的交互数据流应使用不同的传输协议以提高INCS的整体性能. INCS的5种数据流应按如下方式选择传输协议: 采样数据和算法调整信号使用UDP协议; 用户控制信号和系统诊断数据使用TCP协议; 紧急信号使用UDP协议, 而且控制器在确认信号到达前应连续发送.

参考文献(References)

[1] Cardwell N, Savage S, Anderson T. Modeling TCP latency[C]. Proc of IEEE INFOCOM. Israel: IEEE Press, 2000: 1742-1751.

[2] Mellia M, Stoica I, Zhang H. TCP model for short lived Flows[J]. IEEE Communication Letters, 2002, 6(2): 85-87.

[3] Yang Li-man, Li Yun-hua, Yuan Hai-bin. Analysis of time delay in networked control systems and study of data transmission technology[J]. Control and Decision, 2004, l9(4): 361-366.

[4] Wei An, Chen Yuqiang, Wu Jinhua. Simulation study of TCP/IP communication based on networked control systems[C]. World Congress on Intelligent Control and Automation. Dalian, 2006: 4479-4483.

[5] Richard Stevens W. TCP/IP illustrated: The protocols[M]. Boston: Addison-Wesley Longman Publishing, 1993.

[6] Waxman B M. Routing of multipoint connections[J]. IEEE J on Selected Areas in Communications, 1988, 6(9): 1617-1622.

[7] Kenneth L Calvert, Matthew B Doar, Ellen W Zegura. Modeling internet topology[C]. IEEE Communications Magazine, 1997, 35(6): 160-163.

[8] Claffy K Caida. The nature of the beast: Recent traffic measurements from an internet backbone[EB/OL]. http : // www.caida.org / publications / papers / 1998 / Inet98/ Inet98.html.

[9] Vilfredo Federico Damaso Pareto. Pareto efficiency[EB/OL]. http://en.wikipedia.org/wiki/Pareto_efficiency.

[10] Charlie Cleveland. Network simulator-2[EB/OL]. http://sourceforge.net/projects/ns2-linux.

[11] Ken O Burtch. Linux shell scripting with bash[M]. Indiana: Sams Publishing, 2004.

下 期 要 目