# 网络化控制系统中的 TCP/IP 建模研究

韦 安

（中科院自动化研究所复杂系统与智能科学实验室 北京 100080）

**摘 要** TCP/IP 协议已经成为了网络化控制系统中事实上的传输标准协议，但 TCP/IP 协议直接用于网络化控制系统中还存在实时性不足，开销过大等问题，特别是针对网络化控制特殊环境下的 TCP 建模分析还缺乏研究。本文针对网络化控制中的数据流特点，建立了数据传输的 TCP 解析模型，分析了一些重要的参数。实验结果表明，该模型较好地反映了实际网络化控制中的 TCP 传输行为，可以作为分析和改进网络化控制中 TCP 协议的模型。

**关键词** TCP/IP，网络化控制系统，解析模型

## Modeling TCP/IP in Networked Control Systems

WEI An

(The Key Laboratory of Complex Systems and Intelligence Sciences Institute of Automation, Chinese Academy of Sciences, Beijing 100080)

**Abstract** TCP/IP is actually the standard transmission protocol of the networked control systems. But some issues such as real time and cost of the protocol obstacle the application of it in networked control systems. The work on TCP modeling in networked control environment has not been done so much. This paper establishes the TCP analytic model according to character of data flow in networked control systems. Some concerned parameter is analyzed. The experiment result shows that this model is well validated with the real TCP behavior of networked control and can be used as the model to analysis and promote the TCP performance of networked control systems.

**Keywords** TCP/IP, Networked control systems, Analytic model

## 1 引言

基于以太网络技术的网络化控制系统 NCS(Networked Control System)是继现场总线控制系统后工业控制领域发展起来的一种新型控制技术。NCS 系统直接采用以太网作为控制总线，兼容了当前通信技术中的以太网以及网络技术，很好地统一了底层的控制网络协议，建立了良好的兼容接口，因此成为了当前以及以后工业控制领域应用和研究的热点。

完全意义上的 NCS 系统应该是基于 Internet 的控制系统，而不只是基于局域网的现场控制系统，图 1 为典型的网络化控制结构图[1]。因此，NCS 系统可以实现真正意义上的远程控制、监控、故障诊断等以前传统控制系统所无法完成的功能。

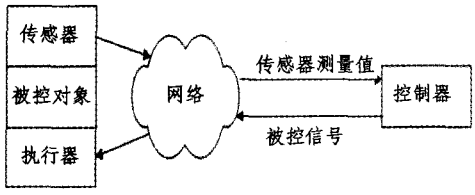


图 1 典型的 NCS 网络结构

实现 NCS 这种基于 Internet 的控制所要面临的一个重要问题是如何实现现场控制网络和外部网络的无缝而又安全的连接。目前在工业应用领域所采用的策略是在 NCS 系统中采用 TCP/IP 协议实现和外部网络的连接。这样的例子有 EtherNet/IP、HSE 工业以太网、浙大中控 EPA 技术等。这些 NCS 系统无一例外地都采用了 Ethernet+TCP/IP 的网络协议。

沿用 TCP/IP 无疑是 NCS 系统的一个优势，但是，TCP/IP 协议真正用于工业以太网实现工业数据的通信所存在的一些深层次的问题还没有得到根本的解决，主要是通信的实时性和可靠性还无法达到工业数据通信的要求。因此需要建立在网络化控制环境下的 TCP 模型，分析影响性能的因素，从而对协议做出改进，提高实时性能，减小协议的开销，更好地满足实时数据通信的要求。

本文分析了网络化控制系统中的数据流特点，根据 TCP 拥塞控制算法，建立了短小数据流下的 TCP 解析模型，主要分析了传输延迟等性能参数，实验结果验证了模型的正确性。

## 2 NCS 中的数据流类型及特征

网络时延是在数据传输过程中产生的，它是否对所有数据都会产生不利影响，这首先需要了解信道传输数据的类型和对实时性的不同要求。

在 NCS 中，需要传输的数据可分为三类：周期数据、猝发数据和非实时数据[2]。

1)周期数据：如各种传感器和控制器的 I/O 信号和部分系统状态监测数据。周期数据对时间有严格要求，一般不允许有秒级的时延，在某些特殊场合下甚至不允许有毫秒级的时延，并且有截至期限制。此外，对周期数据而言，只有最新数据是有意义的，如果在某一时段内，一周期数据由于某种原因未能达到或出错，而此时下一个数据已经产生，则该数据将被丢弃，因此周期数据一般不要求重发。

2)猝发数据：如报警信号和紧急操作指令。猝发数据对实时性要求极高，一般应有比周期数据更高的优先级，并且要求准确无误，但数据长度一般较短，数据量相对较少，对带宽

的占用率较低。

3)非实时数据:如用户编程数据和组态数据。它对时间的要求并不严格,允许有一定的时延,但这类数据的长度较长且不确定,数据量较大,对带宽的占用率较高,一般以小型或微型文件的形式出现。所传送的数据一般都是有意义的,不允许丢失,需要差错控制和重发机制保证数据的完整和准确。

上述三类数据中,周期数据和猝发数据属于实时数据,但对实时性的要求不同,前者应满足截止期要求,对时延特性较为敏感,时延的不确定性影响闭环控制的稳定性和控制性能;后者则希望尽可能快地得到响应,时延越小越好,但并不介意时延是否具有确定性,同时对可靠性地要求高于周期数据。这两类数据有一个共同的特点,那就是数据量小,一般从几个字节到几 k 字节不等,特别是猝发数据,它的最大特点是数据量小,实时性要求高。对于非实时数据,可以通过 http 或者 ftp 协议来传输,相对于前两类数据,这类数据的数据量较大,对带宽的占用率较高,一般在几十 k 字节左右。

## 3 慢启动算法与拥塞避免

当网络中的报文超过网络的处理能力的时候,网络的性能会出现明显的下降,这种现象称为拥塞[3]。为了减小这种拥塞,在 TCP 中广泛地采用了 TCP 拥塞控制算法。这些算法包括慢启动(Slow Start)、拥塞避免(Congestion Avoidance)、快速重传(Fast Retransmit)、快速恢复(Fast Recovery)等,这些算法的使用大大提高了网络传输的性能。

慢启动和拥塞避免算法是发送方用来控制注入网络中的数据量的流量控制策略。

(1)当连接建立之后,使用慢启动算法,发送方拥塞窗口 *cwnd* 的初始值为 1 个报文段长度,慢启动阈值 *ssthresh* 为 65535 字节,随后 *cwnd* 按指数级增长,即:每收到一个 ACK,*cwnd* 增加一个报文段长度。

(2)当拥塞窗口的值达到慢启动阈值 *ssthresh* 的时候,拥塞窗口改为线性增长方式,即:每收到一个 ACK,对拥塞窗口增加 $1/cwnd$。这是拥塞避免。

(3)当新的数据被对方确认时,就增加 *cwnd*,但增加的方法依赖于发送方是正在处于慢启动阶段还是处于拥塞避免阶段。如果 *cwnd* 小于或者等于 *ssthresh*,正在进行慢启动,否则正在进行拥塞避免。

图 2 为 TCP 拥塞控制算法的单条 TCP 链路的拥塞窗口变化曲线。$t=0$,TCP 发送端采用慢启动算法启动,拥塞窗口从其实设定值按照指数规律快速上升,到达慢启动门限后进入线性增长阶段 AI(拥塞避免阶段),窗口值按线性值增加,如果检测到三个以上的 ACK,表明有数据丢失,拥塞控制进入快速重传阶段 FR,重传完成后进入正常的工作阶段即线性增长阶段 AI,如此循环。在以上各阶段中,SS 可以看作是 TCP 起始阶段,AI 是正常工作阶段,其余的阶段都可以看作是非正常工作阶段或者丢包工作阶段。

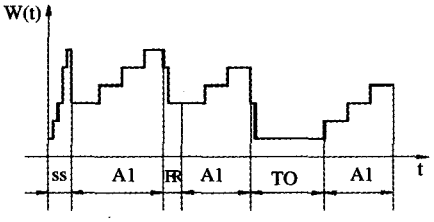


图 2 TCP 拥塞窗口变化曲线

对于短小数据流,从上面的分析可以看出,慢启动阶段和线性增长阶段对拥塞控制的影响和作用更大一些。在窗口较小的情况下,满启动阶段的窗口增加和拥塞避免阶段的窗口增加非常近似(较小窗口下指数增加近似线性增加),并且只有当拥塞窗口值大于 2 个报文段或者 1 个 MSS(最大报文段长度)(RFC2581)的时候拥塞控制才进入线性增长阶段。

在第 2 节中分析了 NCS 中的数据流,所有类型的 NCS 报文段都不会超过几十个 k 字节,而 MSS 的最大长度为 1460Byte,因此在 NCS 系统中,最大报文段不会超过 70MSS。这些报文在没有数据丢失的情况下,绝大部分会在慢启动阶段完成传送任务。

## 4 NCS 中 TCP 模型

从第 3 节中的分析可以得出,网络化控制中数据 TCP 数据传送主要工作在慢启动阶段。下面建立慢启动阶段的 TCP 解析模型。

每一个 TCP 链接都可以把它分解为三个工作阶段,第一个阶段为链接建立阶段,第二个阶段为数据传送阶段,第三个为链接结束阶段。一般而言,相对于前面两个阶段,第三个阶段不会对整个数据传送过程产生影响,因此本文的建模可以分为链接阶段的建模和数据传送阶段的建模两个部分。

### 4.1 假设条件

本文使用与文[4]相类似的网络和 TCP 传输方案,并假定主机使用 TCP Reno 拥塞控制算法。同时在建模时仅考虑由 TCP 传输所产生的等待时间,暂不考虑由于缓冲限制等因素而产生的端点上的延迟。假定发送方能以其拥塞窗口所允许的最快速率发送完整的报文段,而且接收方通告一个固定大小的流控制窗口。在建模中还假定接收方使用 RFC2581 中说明的延迟应答方案,暂不考虑 Nagle 算法和糊涂窗口避免的影响。

本文在建模中使用循环(rounds)来模拟等待时间。一个循环是指从发送方开始一个窗口分组的传输,到发送方收到对一个或多个这些分组的应答为止,假定所有分组在慢启动阶段全部发送完毕,没有数据丢失发生。

最后,假定发送一个窗口中所有分组所用的时间远小于循环时间,并且循环往返的持续时间与窗口的大小无关,传输的数据总量假设为有限值 $N$。

### 4.2 链接建立阶段模型

每个 TCP 链接都是从"三次握手"开始的,图 3 表示的是一个 TCP 建立链接的过程。首先由客户端进程发起主动链接请求,发送带有 SYN 标志的序列号为 $x$ 的报文段,服务器端被动打开链接。当接收到客户端发来的链接请求后,服务器如果不忙,或者允许链接则发送带有 SYN 标志的序列号为 $y$ 的报文断,同时应答序列号为 $x+1$。客户端为了最终确认链接的建立还需要回发一个确认序列号为 $y+1$ 的报文段最确认链接的建立。在链接建立的整个过程中,如果链接建立的任何一方在发送链接请求标志 SYN 的超时时间 $T_c$ 内没有收到对方的 ACK,则超时时间加倍并重发 SYN 报文段。

参考图 3,三次握手过程分解为下面的阶段:

阶段 1:客户端 $i\geqslant 0$ 次发送 SYN 失败,第 $i+1$ 次成功。

阶段 2:服务器端 $j\geqslant 0$ 次发送 SYN/ACK 失败,第 $j+1$ 次发送成功。

阶段 3:客户端接到服务器发送的 SYN/ACK 后,确认发送 ACK 后链接建立,开始发送数据。

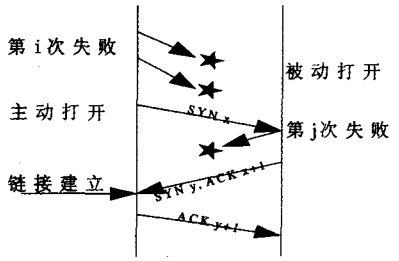


图3 TCP链接建立过程

定义 $P_f$ 为由服务器端到客户端的前向报文丢失率，$P_r$ 为客户端到服务器端的反向报文丢失率，$RTT$ 为报文的平均往返时延。

令 $P_h(i,j)$ 为客户端经历了 $i$ 次发送 SYN 失败，第 $i+1$ 次成功，服务器端经历了 $j$ 次发送 SYN/ACK 失败，第 $j+1$ 次成功这个过程的概率，则：

$$p_h(i,j)=p_r^i\cdot(1-p_r)\cdot p_f^j\cdot(1-p_f) \tag{1}$$

这个过程的时间延迟

$$\begin{aligned}D_h(i,j)&=RTT+\sum_{k=0}^{i-1}2^kT_s+\sum_{k=0}^{j-1}2^kT_s\\&=RTT+(2^i+2^j-2)T_s\end{aligned} \tag{2}$$

$D_h$ 的概率分布，即整个链接建立阶段的延迟时间小于等于时间 $t$ 的概率为：

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

根据(1)和(2)，考虑到实际 TCP 链接在经历 3～6 次尝试后会自动取消链接这一实际情况，链接建立阶段的平均延迟时间：

$$\begin{aligned}E(D_h)&=\sum_{D_h(i,j)\leqslant t}(p_h(i,j)\cdot D_h(i,j))\\&=\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}(p_h(i,j)\cdot D_h(i,j))\\&\approx RTT+T_s\left(\frac{1-p_r}{1-2p_r}+\frac{1-p_f}{1-2p_f}-2\right)\end{aligned} \tag{4}$$

### 4.3 数据传送阶段模型

假设数据传送阶段的传送或者是传送第一个分组，或者是还没有发生数据丢失情况下的分组传送。总之，传输工作在慢启动阶段并根据前面的假设没有数据丢失的情况发生。

定义 $E(T_{ss})$ 为慢启动阶段的平均延迟时间。在慢启动阶段，拥塞窗口 $cwnd$ 或者指数增长到门限值 $W_{max}$，或者发生数据丢失后进入快速重传。根据前面的假设，本文仅考虑前一种情况。定义 $E(d_{ss})$ 为慢启动阶段发送的数据报文段期望，$E(W_{ss})$ 为慢启动结束时的拥塞窗口期望值。显然，如果 $E(W_{ss})\leqslant W_{max}$，则 $E(T_{ss})$ 即为慢启动阶段以指数增长的窗口发送数据的时间，如果 $E(W_{ss})\geqslant W_{max}$，则 $E(T_{ss})$ 包括了以指数规律增长到 $W_{max}$ 之前的发送时间以及以 $W_{max}$ 为窗口大小发送完剩余数据的时间。

首先计算 $E(d_{ss})$，显然 $E(d_{ss})=d$，即在慢启动阶段期望发送 $d$ 个报文段的期望值就是 $d$。

下一步，计算报文段在慢启动阶段的平均延迟时间 $E(T_{ss})$。慢启动阶段，发送端根据拥塞窗口 $cwnd$ 的允许大小发送报文段，接收端则每隔 $b$ 个报文段回发一个 ACK，因此对发送端而言，每次相当于接收到 $cwnd_i/b$ 个 ACK。慢启动算法中，发送端的每收到一个 ACK，拥塞窗口值增加 1。令 $cwnd_i$ 为第 $i$ 次的拥塞窗口值，则有：

$$cwnd_{i+1}=cwnd_i+cwnd_i/b=\alpha cwnd_i \tag{5}$$

其中 $\alpha=(1+1/b)$。

令 $W_1$ 为初始的拥塞窗口值，根据慢启动算法第 $i$ 次结束时发送的数据量为：

$$SSd_i=\sum_{k=0}^{i-1}W_1\alpha^k=W_1\cdot\frac{\alpha^i-1}{\alpha-1} \tag{6}$$

从(6)中，可以求解出发送 $SSd_i$ 的次数 $i$ 为：

$$i=\log_\alpha\left(\frac{SSd_i(\alpha-1)}{W_1}+1\right) \tag{7}$$

由(6)、(7)可以得到慢启动阶段发送 $d$ 个报文段后的拥塞窗口值为：

$$\begin{aligned}W_{ss}(d)&=W_1\cdot\alpha^{i-1}=W_1\cdot\alpha^{\log_\alpha\left(\frac{d(\alpha-1)}{W_1}+1\right)}\cdot\alpha^{-1}\\&=W_1\cdot\left(\frac{d(\alpha-1)}{W_1}+1\right)\cdot\alpha^{-1}\\&=\frac{d(\alpha-1)+W_1}{\alpha}\end{aligned} \tag{8}$$

如果取 $\alpha=1.5$，$W_1=1$ 则：

$$W_{ss}(d)\approx d/3 \tag{9}$$

式(9)说明要达到 $W_{ss}(d)$ 窗口值，需要发送 $3W_{ss}(d)$ 字节的数据。

如果 $E(W_{ss})\geqslant W_{max}$，则拥塞窗口先以指数规律增加到 $W_{max}$，根据式(8)，可以得到当前发送的数据段为：

$$d_1=(\alpha W_{max}-W_1)/(\alpha-1) \tag{10}$$

对应的次数为：

$$r_1=\log_r\left(\frac{W_{max}}{W_1}\right)+1 \tag{11}$$

然后发送窗口再以 $W_{max}$ 发送完剩余的数据，这个过程需要 $r_2$ 次：

$$r_2=(d_{ss}-d_1)/W_{max} \tag{12}$$

综合式(10)、(11)、(12)，可以得到当 $E(W_{ss})\geqslant W_{max}$ 时的平均延迟时间为：

$$E(T_{ss})=RTT\left(\begin{array}{l}\log_r\left(\frac{W_{max}}{W_1}\right)+1+\\\frac{1}{W_{max}}\left(E(d_{ss})-\frac{\alpha W_{max}-W_1}{\alpha-1}\right)\end{array}\right) \tag{13}$$

根据式(7)，可以得到当 $E(W_{ss})\leqslant W_{max}$ 时的平均延迟时间为：

$$E(T_{ss})=RTT\cdot\log_r\left(\frac{E(d_{ss})(\alpha-1)}{W_1}+1\right) \tag{14}$$

## 5 实验结果与分析

### 5.1 实验方法

根据 2 中分析的数据流类型，建立了三种数据包类型来模拟实际网络化控制中的数据流类型，如表 1 所示。

表1 实验数据类型

| 包类型 | 报警数据 | 传感器数据 | 编程数据 |
|---|---|---|---|
| 大小(bytes) | <100 | 1k～10k | >10k |
| 传送方式 | 连续 | 周期 | 连续 |

实验中，采用 Client-Server 结构，服务器位于中科院自动化所，客户机位于天津大学。服务器为 Petium4 PC，运行 Windows 操作系统。客户端为 NetCon 平台，NetCon 平台基于高性能 ARM9 处理器设计，运行嵌入式 Linux 操作系统。

实验步骤如下：

1)利用 socket 编写客户端，服务器端程序，客户端数据包采用表 1 的类型。

(下转第 85 页)

点为 $s$、长度为 $L2$，$L2$ 的起点、终点偏移量均为加密长度 $F$ 的整数倍。

最后将 $L2$ 段明文加密后写入磁盘阵列。

现在讨论一下对可能剩余的 $F-g+x(0<(g-x)<(F,x)$ 0 或 $x=0$)个明文数据进行填充后加密：由于这部分数据的长度小于加密长度 $F$，也就是说，如果 $F$ 为 16 个字节，那么这部分数据最多为 15 个字节，这时需要填充 $(g-x)$ 个数据，填充后进行加密处理，并将 $(g-x)$ 写入文件结束的一个字节。无论需不需要填充，文件都需要增加一个字节保存填充的个数，不填充时为 0，填充时为填充的个数，脱密时根据填充的个数去除填充数据。

**结束语** 刀片加密服务器使用户免除后顾之忧；通过服务器加脱密技术使人们开发、应用因特网的商机成为可能；数据加密技术的应用非常广泛。可以预见，随着时代的发展，刀片加密服务器的加脱密技术将会朝着更快速、更安全、更经济的方向前进。

## 参 考 文 献


1 马琳茹. 基于 Linux 平台加密服务器设计与实现[D]. 北京：中国人民解放军国防科学技术大学，2002

2 谭华，沈建军. PSTN 网络智能优化的思考[J]. 电信网技术，2005(5)：19～22

3 王刚. 网络磁盘阵列结构和数据布局研究[D]. 天津：南开大学，2002

4 乐奕平，王辉球. 我国中小企业电子商务发展新模式[J]. 商业研究，2004(22)：179～181

5 夏恒，徐向阳，李仁发. 一种安全实用的密钥管理系统的研究与实现[J]. 计算机应用与软件，2005(12)：113～115


(上接第 76 页)

2)先运行服务器端程序，再运行客户端程序，利用 sniffer 软件监视来自客户机的所有 TCP 包，客户端记录发送时间，根据 sniffer 得到的接收时间和客户端的发送时间得到每次的传输时间值，统计得到平均延迟时间。

3)利用 sniffer 软件解析 TCP 包，得到窗口值，并统计往返时间($RTT$)，接受端通知最大窗口值($W_{max}$)。利用这些参数代到式(13)，(14)计算出平均延迟时间。

测试中，传感器数据采用周期发送，周期为 2 秒。

### 5.2 实验结果及分析

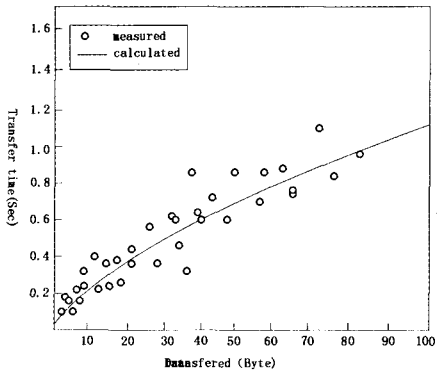


图 4 报警数据传输延迟与模型对比曲线

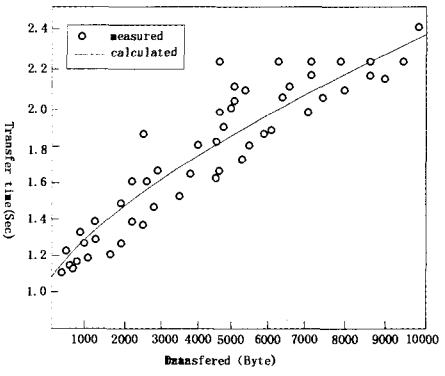


图 5 传感器数据传输延迟与模型对比曲线

图 4～6 分别对应报警数据、传感器数据、编程数据的测量值和模型计算值的曲线。图 4 中测量的数据很好地跟踪了模型曲线，图 5 中数据也较好地跟踪了模型曲线，图 6 中测量数据在较大数据流时，测量数据和模型曲线产生了较大的误差，原因是模型当中没有考虑到慢启动阶段的数据丢失情况，如果数据丢失，则根据图 1，TCP 应该进入快速重传或超时重传阶段，这时执行的拥塞控制算法将不再是慢启动算法，因此影响到了模型的准确性。在下一阶段的建模研究过程中，对于较大数据流将考虑数据丢失情况下的 TCP 模型。总之，可以看出，本文提出的 TCP 延时模型能够较好地反映网络化控制中的主要数据流类型的传输时间，可以作为仿真模型和实际模型使用。

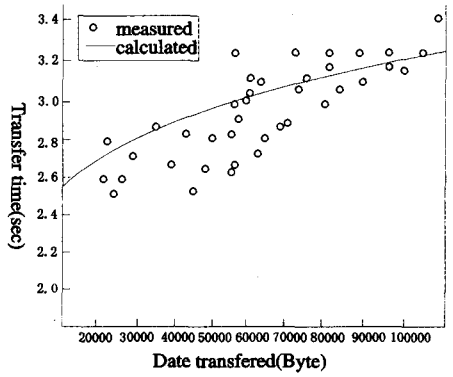


图 6 编程数据传输延迟与模型对比曲线

**结论** 本文从网络化控制系统中数据流特点入手，利用解析的方法建立了数据流传输的延迟模型，通过实验的方法，检验了模型的准确性，同时也指出了模型对长流量数据需要改进的方向。本文给出的模型和方法可以作为研究网络化控制系统中 TCP 建模的的参考模型，为进一步研究网络化控制中的 TCP/IP 通信建立了理论基础。

## 参 考 文 献


1 Walsh G C, Beldiman O, Bushnell L. Error encoding algorithms for networked control systems. In: Proc. of the 38th Conf. on Decision & Control Phoenix, Arizona USA, 1999. 4933～4938

2 杨丽曼，李运华，袁海斌. 网络控制系统的时延分析及数据传输技术研究[J]. 控制与决策，2004，(4)：361～382

3 谢希仁. 计算机网络[M]. 北京：电子工业出版社，2003

4 Cardwell N, Savage S, Anderson T. Modeling TCP Latency. IEEE INFOCOM 2000, Tel Aviv, Israel, 2000. 1724～1751

5 Mellia M, Stoica I, Zhang H. TCP Model for Short Lived Flows. IEEE Communicaion Letters, 6(2): 85～87

6 Pack S, Ahn S, Choi Y. Wireless TCP Model for Short-Lived Flows. IEEE, 1725～1729

7 赵炯，周其刚，张树京. TCP 启动阶段的建模研究. 计算机工程，2003，29(8)：19～21

8 Floyd S, Headerson T. The NewReno modification to TCP's fast recovery algorithm. RFC 2582, April 1999