# 一种提高 XCP 协议在大 RTT 差异环境下的鲁棒性方法

张慧翔　戴冠中　姚　磊　潘文平
（西北工业大学自动化学院　西安 710072）

**摘　要**　现有 TCP 协议的拥塞控制机制存在很多不足，XCP(eXplicit Control Protocol)协议采用显式反馈的方式有效地克服了这些缺陷。XCP 协议能容忍数据流之间一定的 RTT 差异，但当这种差异超出一定范围时，XCP 协议性能恶化，变得不稳定。通过分析 XCP 协议路由控制周期与数据流 RTT 的关系，提出了一种控制周期根据 RTT 差异程度而自适应调整的方法，消除了系统振荡。仿真数据表明，该方法能有效地提高 XCP 协议在数据流 RTT 差异较大的环境下的鲁棒性，同时不给路由器带来过大的计算负担。

**关键词**　显式控制协议，RTT 差异，自适应控制周期，拥塞控制

## Increasing Robustness to RTT Variance for XCP

ZHANG Hui-xiang　DAI Guan-zhong　YAO Lei　PAN Wen-ping
(College of Automation, Northwestern Polytechnical University, Xi'an 710072, China)

**Abstract**　The eXplicit Control Protocol(XCP) was developed to overcome some of the limitations of TCP, such as unclear congestion implication, low utilization in high bandwidth delay product networks, unstable throughput and limited fairness. XCP, however, is robust only for some range of RTT variance. The performance of XCP becomes worse and oscillating while beyond the range. The relationship between router's control interval and flows RTT variance is analyzed, and a method to make the control interval adaptive to the system RTT variance is put forward. By stabilizing the control interval, the revised XCP removes the system oscillation. Simulation results show that the adaptive method increases the robustness to RTT variance for XCP, meanwhile brings little computation load to the XCP router.

**Keywords**　Explicit Control Protocol, RTT variance, Adaptive control interval, Congestion control

## 1　引言

现有 TCP 协议的拥塞控制机制存在很多不足[1]：1)拥塞指示不明确；2)在高延迟带宽乘积网络(High Bandwidth-Delay Product Networks)中效率低下；3)对大 RTT(Round Tri PTime)的数据流带宽分配存在不公平；4)中间节点队列振荡不稳定。针对这些不足，文献[2]提出了一种基于显式反馈的拥塞控制协议 XCP，引起了极大的反响。

XCP 协议引入了一个拥塞头部[4]，见图 1。每个数据包通过拥塞头部携带自身的状态信息，包括当前估计 RTT 值(SRTT)、X(等于 SRTT 除以当前拥塞窗口 cwnd，1/X 就是端系统发送速率)、需求发送速率(Delta_Throughput)和反馈速率(Reverse_Feedback)。路由系统通过拥塞头部可以获取其状态信息。端系统在发送数据时设置其需求发送速率，在数据包传输路径上的路由根据网络状况更新需求发送速率字段，直接控制端系统的拥塞窗口大小。数据接收端拷贝包头中的需求发送速率字段到反馈速率字段，反馈回发送端。发送端根据反馈速率字段更新其拥塞窗口。

XCP 路由对经过的每个数据包进行观测，统计每个控制周期的链路数据流量 $input\_bw$ 和持久队列长度 $q$。XCP 路由系统提供了两个控制器：效率控制器和公平控制器。效率控制器每隔一个控制周期重新计算空闲带宽，公平控制器根据加增乘减(Additive Increase Multiplicative Decrease)[5]的原则分配空闲带宽。当空闲带宽为负值就意味着减少端系统的拥塞窗口大小。XCP 路由效率控制器采用公式(1)[2]计算空余带宽 $F$：

$$F=-\alpha(input\_bw-C)-\beta\frac{q}{d} \tag{1}$$

其中 $C$ 为链路带宽，$d$ 为控制周期；$\alpha$，$\beta$ 为系统参数。当 $F$ 趋于 0 的时候，为了对新加入的数据流分配带宽，在每个控制周期都有 $\gamma \cdot input\_bw$ 的 Shuffling Bandwidth[2]被重新分配。Shuffling Bandwidth 保证了 XCP 协议的公平性。

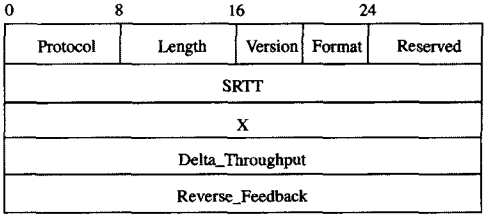


图 1　XCP 拥塞控制包头结构

XCP 协议的基本时间单位是 RTT，其路由效率控制器的采用数据流的平均 RTT 值作为控制周期。文献[2,3]通过大量仿真实验表明当所有数据流具有相同的 RTT 值时，效率控制器具有很好的效果，系统具有很好的稳定性。文献[6]从理论上分析当数据流 RTT 处于[10ms, 500ms]时，系统仍表现出较好的性能，XCP 对数据流 RTT 差异仍具有较好的鲁棒性。当数据流 RTT 的差异超出该范围时，系统变得不稳定，路由队列出现大幅振荡，端系统拥塞窗口出现剧烈振荡(见图 3、图 5)。

**张慧翔**　博士研究生，研究方向为 Internet 网的拥塞控制和 QoS 控制；**戴冠中**　教授，博士生导师，研究方向为自动控制、信息安全；**姚　磊**　博士研究生，研究方向为网络分布式系统；**潘文平**　博士研究生，研究方向为 Web QoS 控制。

## 2 相关工作

在数据流 RTT 差异大的环境下，采用平均 RTT 作为控制周期是不合适的。文献[3]建议采用有限分组的方式来提高系统对 RTT 差异的鲁棒性。比如可以约定 $RTT<500ms$ 为第 1 组，$500ms<RTT<1000ms$ 为第 2 组，$1000ms<RTT<1500ms$ 为第 3 组。每个分组具有独立的效率和公平控制器。每个控制组享有的带宽根据其数据流数目来分配。每组的效率和公平控制器使组内的数据流收敛到组内公平。这种方案有几个缺陷：一是 XCP 路由并不维护数据流状态，每组数据流数目采用 $N=\sum\frac{s_i}{d\times r_i}$ 计算[3]，这种计算对 $N$ 恒定情况比较准确，对变化的 $N$ 估算有较大误差，造成每个分组应该享有的带宽不准确，直接导致队列剧烈振荡。二是每个控制分组之间带宽共享机制实现比较复杂，整个系统达到最大最小公平是一个难题。三是对于每个分组临界处数据流处理不好，一个 RTT 为 490ms 的数据流其估算的 RTT 值可能会大于 500ms，那么其就在分组 1 和分组 2 之间波动，造成分组数据流数 $N$ 估计的误差。

本文通过分析 XCP 路由控制周期与数据流 RTT 的关系，提出了一种自适应调整控制周期的方法。仿真数据表明，该方法能有效地提高 XCP 协议对数据流 RTT 差异的鲁棒性，同时不给路由带来较大的计算负担。文章组织如下，第 3 部分描述了 RTT 差异给系统带来的不稳定现象，分析了数据流 RTT 与控制周期之间的关系，给出了控制周期自适应的原则。第 4 部分给出了控制周期自适应调整的方案，并和 XCP 协议进行了对比仿真实验。最后总结了我们的工作和下一步的目标。

## 3 RTT 与控制周期的关系

XCP 路由根据所有数据包的 RTT 值来计算其下一个控制周期[4]：

$$sum\_x=\sum X_i=\sum\frac{srtt_i}{cwnd_i} \tag{2}$$

$$sum\_x\_rtt=\sum X_i\cdot srtt_i=\sum\frac{srtt_i^2}{cwnd_i} \tag{3}$$

在每个控制周期计算新的控制周期 $d$：

$$d=\frac{sum\_x\_rtt}{sum\_x}=\frac{\sum_{i=1}^{N}d_i}{N} \tag{4}$$

$X$ 相当于一个权值系数，通过这种方法计算的平均 RTT 是从数据流的层次来平均的，拥塞窗口越大权值系数也就越小。

### 3.1 系统稳定时的控制周期

考虑图 2 中单瓶颈的网络拓扑。如果系统之中存在 $N$ 个数据流，所有数据流具有相同的传播延迟 $delay$，当系统稳定时，瓶颈队长 $q=0$，瓶颈链路数据流量 $input\_bw=C$，每个数据流的吞吐量相等，即每个数据流具有相同的 $cwnd$。$srtt=delay+delay\_q+delay\_c$，包括了传播延迟 $delay$，排队延迟 $delay\_q$ 和处理延迟 $delay\_c$，我们不考虑数据包的处理延迟，在稳定状态下有 $srtt=delay$。在每个控制周期由公式(2)，(3)，(4)可得：

$$d=\frac{sum\_x\_rtt}{sum\_x}=\frac{\sum srtt_i^2}{\sum cwnd_i}=srtt=delay \tag{5}$$

式(5)表明当系统稳定时，控制周期为 $srtt$。

XCP 是一种基于窗口的协议，该类协议具有突发性的特点[10]。在数据流进入系统的起始阶段，大量数据被同时发送到网络，会导致数据包排队，$delay\_q$ 不可忽略，此时 $d=srtt>delay$，也就是说控制周期在数据流加入系统的起始阶段会出现突增的现象。然后随着队列数据排空稳定到数据流平均 RTT。

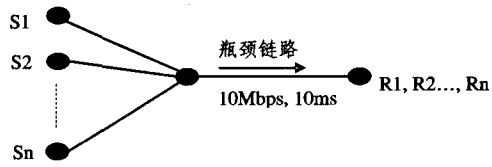


图 2 网络仿真拓扑

### 3.2 小 RTT 数据流加入对控制周期的影响

假定系统带宽被一大 RTT 数据流占据。系统稳定后，$d=srtt1$，$srtt1$ 为大 RTT 数据流估计 RTT 值，网络处于高负载。有一 RTT 较小的数据流加入系统，其 RTT 估计值为 $srtt2$。假定 $\frac{srtt1}{srtt2}=m>1$，那么在 $d$ 时间内，小 RTT 数据流完成了 $m$ 个 RTT 周期，相当于有 $m$ 个小 RTT 数据流加入到系统中。如果不考虑排队延迟，根据公式(2)，(3)，(4)有

$$sum\_x=\sum_{i=1}^{m}srtt2_i+srtt1=m\cdot srtt2+srtt1$$

$$sum\_x\_rtt=\sum_{i=1}^{m}srtt2_i^2+srtt1^2=m\cdot srtt2^2+srtt1^2$$

$$d=\frac{sum\_x\_rtt}{sum\_x}=\frac{srtt1+srtt2}{2} \tag{6}$$

式(6)表明小 RTT 加入系统后，控制周期变为两个数据流的平均值，与两者 RTT 的差异 $m$ 无关，而且这种变化在一个控制周期内就可完成，不会产生控制周期的振荡。如果考虑新数据流加入产生的排队延迟，系统可能需要多个控制周期稳定到 $\frac{srtt1+srtt2}{2}$。

网络处于高负载，空余带宽 $F=0$，新加入的数据流获取带宽只能依靠 Shuffle Bandwidth。大控制周期意味着单位时间内重新分配的 Shuffle Bandwidth 较少，新加入的数据流收敛较慢；但是由于小 RTT 流在相同的吞吐量下所需的 $cwnd$ 较小，大控制周期对小 RTT 流收敛影响不明显。

### 3.3 大 RTT 数据流加入对控制周期的影响

假定系统带宽被一小 RTT 数据流占据。系统稳定后，有一大 RTT 数据流加入系统。XCP 设置初始拥塞窗口为 1 到 4 个数据包大小[8,9]。由于大 RTT 数据包传播延迟长，路由的控制周期较小，路由在很多控制周期会观察不到大 RTT 数据包，控制周期更新近似等于小 RTT。在能够观察到大 RTT 数据包的周期，控制周期更新近似等于两者的均值。这样，控制周期产生了振荡，使系统的性能恶化。两者 RTT 差异严重影响系统性能，小控制周期对大 RTT 数据流收敛的影响可能是致命的。

### 3.4 XCP 协议在 RTT 差异较大情况下的性能

通过 ns2[7] 网络仿真软件观察 XCP 协议在 RTT 差异较大的情况下的性能。网络拓扑结构采用文献[2]中的三角形拓扑，见图 2。XCP 参数设置为文献[2]中推荐参数($\alpha=0.1$，$\beta=0.226$，$\gamma=0.1$)，系统最大 RTT 改为 2s。实验观察了端系统的拥塞窗口和吞吐量以及路由的控制周期和队列变化情况。

仿真 1：两个数据流端系统和路由的带宽都为 10Mbps，瓶颈链路延迟为 10ms，边界链路延迟分别为 5ms 和 490ms。两个数据流的 RTT 至少为 30ms 和 1000ms。仿真结果见图 3，flow1 代表小 RTT 数据流。可以看出由于数据流 RTT 的差异带来了明显的系统振荡。小 RTT 流拥塞窗口更新快，在网络低负载时其能快速获得带宽。而大 RTT 流拥塞窗口

更新慢，相当于在小 RTT 流占据系统带宽时加入系统。

仿真 2：假设系统带宽被小 RTT 流占据，有一大 RTT 流加入。9 个边界链路延迟为 5ms 的数据流从 0 时刻开始，迅速占据路由带宽，在 20s 时刻，边界链路延迟为 490ms 的数据流加入系统。仿真结果见图 5，flow1 代表了 9 个小 RTT 数据流。从图中可以看到控制周期出现剧烈振荡，大 RTT 数据流拥塞窗口一直处于振荡，无法收敛。

### 3.5 设计启示

由于数据流之间较大的 RTT 差异，使控制周期振荡导致了系统的不稳定。大控制周期对小 RTT 流性能影响不大，小控制周期可能会严重损害大 RTT 流的收敛。基于以上分析，确定了两个原则来改善由于 RTT 差异带来的系统性能恶化的问题：

1）系统 RTT 差异越大，控制周期越偏向大 RTT，大 RTT 数据流对系统影响大。

2）控制周期振荡时，控制周期趋向于稳定在大 RTT，振荡不利于系统的收敛。

## 4 控制周期自适应调整的方案

基于 3.5 节中的两个设计原则，提出了一种控制周期自适应的改进方案，使 XCP 协议能适应更大范围的 RTT 差异。

首先对每个数据包根据其 RTT 值进行分组，比如 0～500ms 为分组 1，500～1000ms 为分组 2，1000～1500ms 为分组 3。按照 XCP 路由平均 RTT 的计算方法，对每个数据包计算其所在组的 *sum_x* 和 *sum_x_rtt*。

路由系统加入一个 RTT 差异定时器定时计算系统的 RTT 差异程度。定时器周期取系统容忍最大 RTT 值，比如 2.0s，使得该定时器能观测到数据流在一个完整 RTT 内流经路由的所有数据包。在 RTT 差异定时器的每个周期，利用 3 个分组的 *sum_x* 和 *sum_x_rtt* 计算每个分组的平均 RTT。设三个平均 RTT 值中最大为 *avg*_max，最小为 *avg*_min，平均 RTT 为 0 的分组不参与计算。定义系统 RTT 差异度 *RTT_diff* 来衡量数据流 RTT 的差异程度，计算公式为

$$RTT\_diff = 1 - \frac{avg\_\min}{avg\_\max} \tag{7}$$

当三个分组只有一个分组大于 0 或者全部为 0 时，*RTT_diff* 取 0。

在 RTT 差异定时器的每个周期同时根据三个分组平均 RTT 计算加权平均 RTT，记为 *RTT_w*。表 1 描述了根据三个分组的平均 RTT 值计算 *RTT_w* 采用的权值。分组模式指明了每个分组平均 RTT 是否有大于 0 的值，如'101'就表示分组 2 的平均 RTT 为 0，分组 2 不参与 *RTT_diff* 的计算。

XCP 路由在效率控制器的控制周期根据 *RTT_diff* 更新其控制周期，当 RTT 差异度大于 0.5 时，控制周期取 *RTT_w*，否则取 XCP 协议原有的平均 RTT 值。

表 1 *RTT_w* 计算权值系数

| 分组模式 | 分组 1 | 分组 2 | 分组 3 |
|---|---|---|---|
| 111 | 0.1 | 0.4 | 0.5 |
| 110 | 0.2 | 0.8 | ——— |
| 101 | 0.1 | ——— | 0.9 |
| 011 | ——— | 0.3 | 0.7 |
| 其它组合 | 1 | 1 | 1 |

## 5 仿真实验

采用 3.4 节的网络拓扑和系统参数，对改进的 XCP 协议进行了仿真。设置 RTT 差异定时器周期为 2s。

仿真 3：采用改进的协议重复仿真 1，结果见图 4。和图 3 比较发现，由于控制周期稳定在大周期，端系统拥塞窗口不再振荡，吞吐量比 XCP 更为稳定。路由队列基本没有振荡，呈现缓慢收敛的趋势。

XCP 路由效率控制器采用公式(1)计算空余带宽。准确计算空余带宽需要得到正确的链路数据流量并及时反映队列长度。只要 XCP 路由在其一个控制周期内可以观察到所有端系统一个 RTT 时间内的数据流量，就能计算出正确的链路数据流量。在大控制周期下路由控制器是能观测到小 RTT 数据流一个完整 RTT 内的流量的。但是控制周期过长，在计算新的空余带宽时采用的 $q$ 值不能反映网络负载变化，路由不能及时清空队列中排队的数据包，会导致队列短时期内振荡。由于 XCP 协议本身的收敛[2]，队长基本上是变小的趋势。

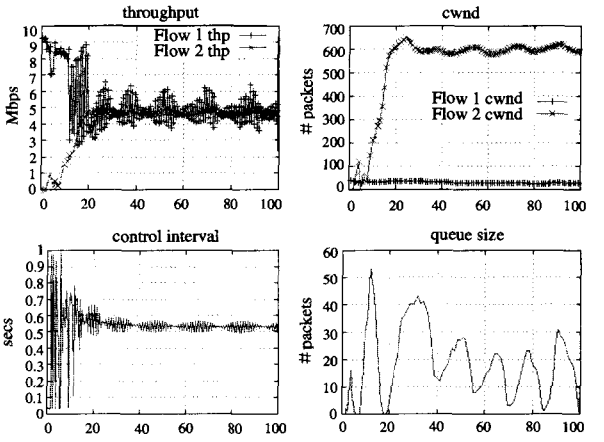


图 3 XCP 协议在大 RTT 差异时的性能

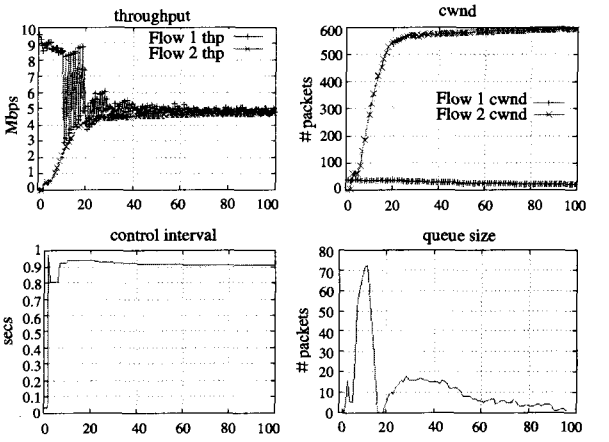


图 4 改进后 XCP 协议在大 RTT 差异时的性能

仿真 4：采用改进的协议重复仿真 2，结果见图 6。图 5 的仿真结果表明原有 XCP 协议在大 RTT 数据流加入系统时，性能很差。和图 5 比较，路由从小控制周期平滑过渡到一个偏向于大 RTT 的控制周期，基本消除控制周期的振荡。大 RTT 流能够较快地收敛到其享有的 1Mbps 的带宽，性能得到明显改善。

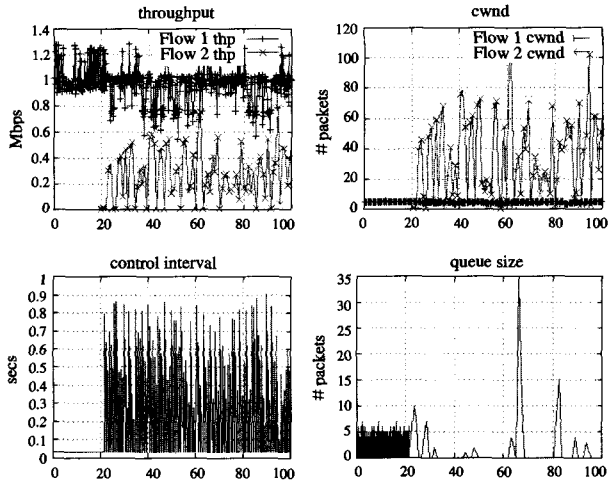


图 5　大 RTT 数据流加入系统的性能

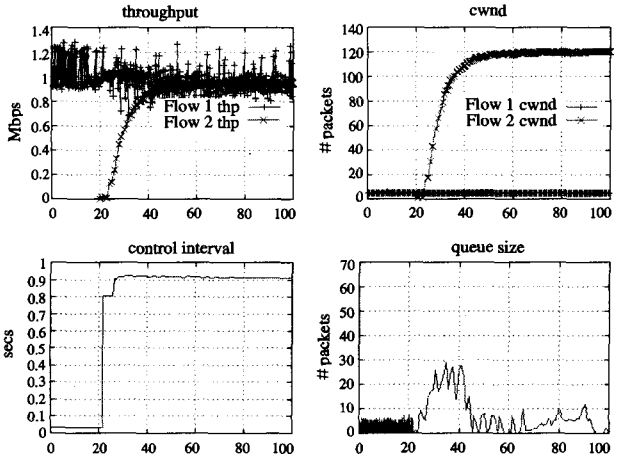


图 6　改进后大 RTT 数据流加入系统的性能

**结束语**　现有 XCP 协议对数据流的 RTT 差异有一定的鲁棒性，当数据流差异比较大时会导致系统性能恶化。本文通过分析数据 RTT 和控制周期之间的关系，提出了一种基于加权平均的自适应调整控制周期的改进方案。该方法没有给路由系统带来更多的计算负担，有效提高了系统对 RTT 差异的鲁棒性。分析表明，Shuffle Bandwidth 在网络高负载时决定着系统收敛的速度。我们将继续研究通过修改 Shuffle Bandwidth 的计算方法来提高系统在网络高负载时对新加入数据流的反应时间。

## 参 考 文 献


[1] 张慧翔，戴冠中，等. 具有显示反馈的拥塞控制系统研究进展. 计算机科学(已录用)

[2] Katabi D, Handley M, Rohrs C. Congestion control for high bandwidth-delay product networks. ACM SIGCOMM Computer Communications Review, 32(4): 89-102

[3] Katabi D. Decoupling Congestion Control and Bandwidth Allocation Policy with Application to High Bandwidth-Delay Product Networks. Ph. D. Dissertation. Massechussetts Institute of Technology, March 2003

[4] Falk A, Katabi D, Pryadkin Y. Specification for the Explicit Control Protocol (XCP). draft-falk-xcp-02. txt (work in progress), November 2006

[5] Chiu, Jain. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. Journal of Computer Networks and ISDN, 17(1): 1-14

[6] Balakrishnan H, Dukkipati N, McKeown N, et al. Stability Analysis of Explicit Congestion Control Protocols. Stanford University Department of Aeronautics and Astronautics Report: SUDAAR 776, September 2005

[7] The network simulator ns-2. 30. http://www.isi.edu/nsnam/ns

[8] Allman M, Paxson V, Stevens W. TCP Congestion Control. RFC 2581. April 1999

[9] Allman M, Floyd S, Partridge C. Increasing TCP's Initial Window. RFC 3390, October 2002

[10] Welzl M. Network Congestion Control. John Wiley & Sons Ltd, 2005


(上接第 24 页)

该套接口占用的其它资源。

### 5.2　数据重发机制

数据重发机制是利用相应的缓存数据来保证应用数据更加准确无误地到达目的地的。send 调用出错时，有两种可能：

一是对等端(IP 地址与端口)已经异常关闭，此时已无法重发，缓存的数据也没有任何用处，此时的主要操作有：

(1)清空文件描述符指示的 list 容器的内容；

(2)发送客户端/服务器已经关闭写的类型包；

(3)从读/写事件中清除该文件描述符；

(4)根据 4.4 判断规则使用 close 或 shutdown 调用关闭 socket 连接。

二是发送缓冲区已满或剩余空间太小不足以容下缓存中数据块，因此要保存未成功发送的数据等待重发。重发的次数由系统全局变量(SENDTIMES)设定，每个连接的 CACHE 初始化时，都要把它的成员变量 ttl 设置为 SENDTIMES。发送每出现一次非 EWOULDBLOCK 错误时，ttl 自减 1。当 ttl 等于 0 时，则认为对等端已无法正确接收任何数据，可以进行连接的关闭和资源回收操作了。

**结束语**　本文研究了采用 I/O 复用技术来实现网络隔离系统中应用代理的方法。使用基于 I/O 复用技术比采用其他技术有着更明显的优势，主要表现在：首先由文中分析可知，I/O 复用技术更适合于网络隔离系统的代理模型；其次，I/O 复用技术避免了使用多线程(进程)时系统进行线程(进程)切换时带来的时间开销，同时也比多线程(进程)方案容易控制；再次，I/O 复用技术把读写事件，接收连接和发起连接等过程交给内核处理，效率上有着显著的提高；最后，使用 I/O 复用传输多路数据的方案更加容易集成其他安全策略，如内容关键字检索、头域关键字过滤以及其他访问控制策略等。因此，从易操作性、安全性、可控制性以及运行效率等方面分析可知，基于 I/O 复用技术实现的代理更适合于网络隔离系统。

## 参 考 文 献


[1] GB/T 20279-2006. 信息安全技术网络和终端设备隔离部件安全技术要求

[2] GB/T 20277-2006. 信息安全技术网络和终端设备隔离部件测试评价方法

[3] Fung K P, Chang R K C. A Transport-level Proxy for Secure Multimedia Streams. IEEE Internet Computing, November/December 2000

[4] Lindskog S, Grinnemo K-J, Brunstrom A. Data Protection Based on Physical Separation Concepts and Application Scenarios // Gervasi O, et al., eds. ICCSA 2005, LNCS 3483, 2005: 1331-1340

[5] Stevens W R, Fenner B, Rudoff A M. Unix Network Programming // Third Edition. Volume 1, The Sockets Networking API. Addison-Wesley, 2004

[6] Johnson M K, Troan E W. Linux Application Development. Second Edition. Addison-Wesley, 2005