单客户机-多服务器模式下IOCP的应用与研究

王 震[1] 徐 博[2] 解永平[1] 孙加奉[1]

(大连理工大学信息与通信工程学院 大连 116024)[1] (中国移动大连分公司 大连 116021)[2]

摘 要 对特定网络应用中的单客户机-多服务器模型进行了分析,给出了基于IOCP的客户机基本实现,又给出了线程调度、异步连接等IOCP框架中的实现方法。针对不定长数据接收的要求,对内存池管理、缓冲区设置等问题进行了分析与设计。连接测试和吞吐能力测试的结果表明,本设计具备高效较大数量连接和数据处理的能力。通过实验方式给出了IOCP线程数设置的建议。

关键词 单客户机-多服务器,完成端口,多线程,内存池,缓冲区

中图法分类号 TP393 **文献标识码** A

Application and Study of IOCP in Single Client-multi Server Model

WANG Zhen[1] XU Bo[2] XIE Yong-ping[1] SUN Jia-feng[1]

(School of Information and Communication Engineering, Dalian University of Technology, Dalian 116024, China)[1]

(Dalian Branch of China Mobile, Dalian 116021, China)[2]

Abstract This paper analyzed a single client - multi-server model based on specific network applications. This paper not only gave the client basic implementation based on IOCP, but also gave the thread scheduling, asynchronous connections IOCP framework method. Aiming at receiving variable length data for the request, memory pool management and buffer settings were analyzed and designed. By connecting test and throughput test, results show that this design has capabilities of a large number of connections and efficient data processing. The recommendations of the number setting of IOCP worker threads were given by the experimental methods.

Keywords Single client-multi server, IOCP, Multi-thread, Memory Pool, Buffer

1 引言

在基于网络通信的设计方案中,大多采用客户机/服务器模型,即一台或多台服务器为大量的客户机服务,也称为多客户机-单服务器模式。然而,在某些特定的应用场合,例如对服务器的故障监控或数据获取,通常是一台或多台客户机监控大量的服务器,即单客户机-多服务器模式。

无论是多客户机-单服务器模式,还是单客户机-多服务器模式,其核心是解决一对多的通信并发处理问题。为了解决网络并发模式的问题,传统上采用一个线程建立一条连接,这就是WinSock IO模型中的雏形——blocking模型。随着实际需求的不断提高,网络连接的数量越来越大,这种独占线程资源的方式逐渐被先进的方法所取代[1]。在Windows NT3.5中引入了IOCP(I/O Completion Port)管理内核对象,并且作为一种网络编程模型引入到Winsock2中。IOCP是迄今为止,最高效的Windows下处理并发网络事件的IO模型。

2 相关工作

在实际应用中,IOCP大多被应用于服务器端,现有的大量文献都对IOCP作为服务器端的底层通信方案进行了详细的讨论。文献[2]对比了网络通信中不采用IOCP的串行方式和采用IOCP的并行方式,并给出了IOCP的服务器端的基本实现。但是,对于IOCP的使用优化上做的稍显不足。文献[3]在构建IOCP服务器端框架的基础上,加入了异步接受连接的处理机制,并且对于底层通信进行了模块化的划分。不过,由于模型的应用背景的缘故,其使用范围受到了局限。此种模型适合于底层通信量比较小的应用,处理效率上佳。文献[4]在IOCP服务器框架的基础上,给出了比较完整的优化处理方案。其核心技术中的线程池的使用增强了系统的可扩展性,对象池的使用增强了软件实时处理数据的能力,同时加强了鲁棒性。但是,该模型对于缓冲区的设置上仍然采用传统的环形队列缓冲方式,对于不定长的数据接收处理略显不足,并且文中没有对接收之后的数据处理给出一种通用的设计方案。文献[5-7]分别在系统性能测试、智能交通监控、远端图像采集等应用领域使用了IOCP模型,但都是将IOCP应用于服务器端。

在特定的应用需求下,客户机需要连接大量的服务器,此时就需要在客户机上使用IOCP。本文在IOCP基本框架的基础上,研究并给出了IOCP客户机实现方法,并且针对不定

王 震(1986—),男,硕士生,主要研究方向为计算机网络通信,E-mail:wangzhen_dlut@mail.dlut.edu.cn;徐 博(1981—),男,硕士,助理工程师,主要研究方向为移动通信;解永平(1966—),男,硕士,副教授,主要研究方向为传感器网络、智能仪表、网络通信,E-mail:xieyp@dlut.edu.cn(通信作者)。

长大数据量的接收缓冲等问题进行了重点讨论。

3 IOCP 原理

与 Windows 早些时候的 Winsock IO 模型相比，IOCP 模型提供了最大的可扩展性，这种模型适合于同时处理的 Socket 数量可以达到几十万个。IOCP 模型中，操作系统会把预先定义好的重叠 IO 请求的通知放入消息队列中，并预先创建好一定数量的工作者线程来处理这些通知。当某一个 Socket 上的 IO 事件完成之后，操作系统会通知一个 IOCP 工作者线程，并在该线程里对数据进行处理。在 IOCP 的理想模型中，每个 IOCP 工作者线程都可以从系统获得一个 CPU "时间片"，轮番运行并检查 IOCP。由于 IOCP 将事件通知和 IO 处理相互隔离，因此，对于同时处理大量连接的系统，IOCP 模型的优越性就非常显著[8-10]。

4 系统方案设计及实现

4.1 系统分析

通常情况下，服务器端需要监听大量客户机的连接请求，并与之建立连接，通过 Socket 进行 IO 通信。然而，在某些特定的场合(比如，对数量庞大的大型设备进行监控的情况下，设备通常作为服务器出现在网络中)，情况则恰恰相反，需要客户机同时连接大量的服务器，如图 1 所示。客户机在与大量的服务器建立连接时，需要从服务器获取相应的数据，由于具体应用背景的不同，获取数据的长度是不定长的，少则几个字节，多则上百 k 字节。

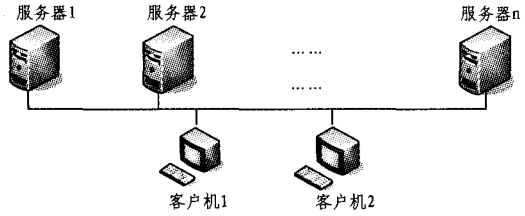

图 1 单客户机-多服务器模式示意图

本系统的性能指标如下：(1)服务器最大的连接数量：200；(2)单条数据长度：不小于 256 字节，最大 100k 字节；(3)每秒最大数据条数：500。

上述单客户机-多服务器的模式，虽然在形式上与传统的单服务器-多客户机模式不同，但是，深入到 TCP 网络底层通信机制可以看出，一旦 TCP/IP 建立了连接，其本质都是一对多的连接，关键技术都在于网络通信的高效并发处理。因此，使用于服务器端的网络 I/O 模型可以应用于本设计中。

针对需要连接大量服务器的性能要求，本设计在 Windows 平台上开发并行网络通信时，采用 Windows 下高效的 IOCP 模型作为核心通信框架，如图 2 所示。在图 2 中，IOCP 对象属于操作系统内核对象，通过设置一定数量的 IOCP 工作者线程，将 Socket 关联到 IOCP 对象并投递 IO 操作请求，IOCP 会异步处理进程发出的 IO 请求，并且将 IO 结果在 IOCP 工作者线程中返回。由此看出，IOCP 完成的工作是底层的网络数据操作，为了能充分发挥 IOCP 高效的底层网络 IO 性能，必须处理好两个关键点：其一是 IOCP 工作者线程的调度，其二是接收之后的数据缓冲设置，即内存管理。图 2 中的对象管理、线程管理、数据管理中都涉及到缓冲区的数据处理问题。这两个核心问题的解决将是决定 IOCP 处理并发数据传输稳定、可靠的关键。

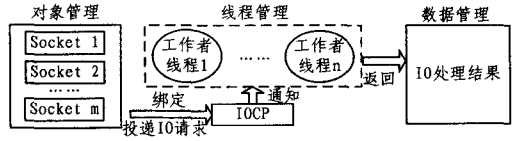

图 2 IOCP 使用框架

本质上，数据在内存中的管理方式有 2 种。一种是传值，即根据需要将数据复制到指定的内存区域中；另一种是传地址，即先把数据复制到一个指定的内存区域里，在接下来的处理环节中采用传递内存首地址的形式。通过上述分析，可以看出，采用传地址的方式比传值的方式减少了数据复制次数，这对于实时性要求比较高的应用环境下是至关重要的。

本设计的特定要求是接收的数据不定长，即在数据接收与数据处理之间需要有一个组包的过程。由于无法估计每次通信包的数量，因此需要有一种机制，能够对每一条连接所接收到的数据包进行缓存。

4.2 系统整体构架设计

针对以上的系统需求及技术分析，系统基于 IOCP 模型的设计方案如图 3 所示。

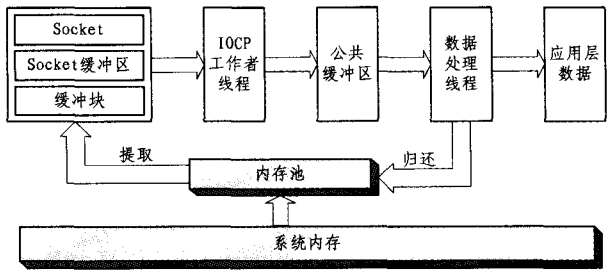

图 3 基于 IOCP 的客户机设计方案

内存管理采用内存池技术。预先申请一定数量固定大小的内存块，由内存管理对象统一进行管理。接收到的每包数据放在内存块中，采用链表的形式将多个内存块进行连接起到数据缓存的效果。即内存由内存池单独管理，负责内存的分配、回收、循环利用，在内存的使用过程中采用传地址的方式对缓冲区进行管理。这样，使系统的稳定性和实时性得到提高。

IOCP 工作者线程(多个)用于处理处于连接状态的 Socket 上的所有的 IO 事件，每次接收的数据存于从内存池中提取来的内存块中。根据应用层的要求，一组通信数据接收完毕之后，从对应的 Socket 缓冲区(每个 Socket 对应于一个缓冲区)中整体传地址到公共缓冲区。

数据处理线程(一个)定时查看公共缓冲区，用于对公共缓冲区数据进行提取、处理，得到应用层数据。

本设计基本工作流程是：

(1)初始化 IOCP 环境(主线程)；

(2)发起连接(主线程或连接线程)，采用异步连接机制；

(3)接收网络数据(IOCP 线程)，由于数据的不定长，引入公共缓冲区和数据处理线程；

(4)完整数据处理(数据处理线程)，完成与应用层有关的操作。

4.3 内存管理——缓冲区内存池

4.3.1 内存池管理

内存池管理要实现的功能有：

(1) 初始化时预留一定数量的内存单元；

(2) 使用时提取。若内存池为空，立即批量动态申请；

(3) 使用结束归还。

动态申请的内存单元要统一为固定大小。这是因为大小不一地动态申请内存，会使堆上的内存愈加杂乱，内存块的大小愈是大小不一，操作系统找出一块合适区域的时间就会越长，即系统的额外开销会越大。因此，在设计中，将内存块的大小设计成统一尺寸，可以解决上述问题，这是一种以固定尺寸来减轻系统负担的方法[11]。

内存管理模式框图如图 4 所示。

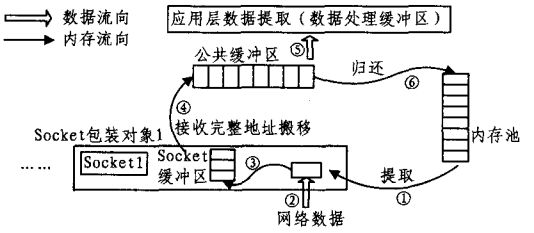

图 4　内存管理模式

4.3.2　缓冲区设置

本设计所使用的缓冲区划分为：Socket 缓冲区；公共缓冲区；数据处理缓冲区。

(1) Socket 缓冲区——为每一个连接建立一个缓冲区，即每一个 Socket 对应于一个缓冲区，方便为每一个 Socket 缓存数据，在接收到应用层的结束标志之前，数据存放于 Socket 缓冲区。Socket 缓冲区采用链表的形式，其中的每一个节点都是以堆形式存在的固定大小的内存块。

(2)公共缓冲区——一旦 IOCP 工作者线程检测到应用层的结束标志，就会将 Socket 缓冲区中的内存块地址依次传递到公共缓冲区队列中。

(3)数据处理缓冲区——数据处理缓冲区存在于数据处理线程内部，其本质是以栈的形式存在于线程内部的内存栈空间。从公共缓冲区读取的数据以传递值的方式复制到数据处理缓冲区，之后，再进行应用层的数据提取。

本设计中采用的 Socket 缓冲区和公共缓冲区都是在逻辑上的划分，没有物理上的划分。缓冲区实体均采用固定大小的内存堆，放在内存池中统一管理。Socket 缓冲区和公共缓冲区都是以单向链表的形式存在，只是一个逻辑的划分，当有数据接收的时候，Socket 缓冲区链表会从内存池中取出内存单元，存放数据，一旦检测到结束标志，就将其链表中的全部节点地址依次搬移至公共缓冲区队列，也即是说，实际传递的每个内存块的地址，不是通过通常的复制值的形式传递的。

4.4　线程调度

4.4.1　IOCP 工作者线程

本设计中，IOCP 工作者线程处理基本的事件返回，要实现以下功能：

一旦返回，就需要判断 IO 事件的状态。其分为连接返回、接收返回、发送返回。由于本设计主要应用于监控，对于连接返回和接收返回重点讨论。

(1) 连接返回

一旦 GetQueuedCompletionStatus() 函数成功返回，并且从单 IO 节点信息中判断为连接返回，说明此时 TCP/IP 连接已经建立，所要做的就是与应用层有关的一系列处理，比如输入用户名、密码、指令等操作。

(2) 接收返回

接收数据返回时，应用程序需要进行以下三个操作：预处理；组包；将内存块地址从 Socket 临时缓冲区搬至公共缓冲区。

在自定义的重叠 IO 结构体中，需要一个指针指向每次数据接收缓冲区的首地址，每次投递 IO 接收请求时，需要从内存池中提取一个内存块并将该指针指向该内存块。预处理的功能之一就是将装载本次接收数据的内存块添加至 Socket 缓冲区的链表尾部。对应于不同的应用层数据，都会有结束标志字符。当本包中没有结束标志时，Socket 缓冲区会在尾部堆积。一旦检测到结束标志，程序会将 Socket 缓冲区的数据一并添加至公共缓冲区队列尾部。图 4 中的步骤①②③④是在 IOCP 工作者线程中完成的。

4.4.2　数据处理线程

数据处理线程的功能包括：

(1) 从公共缓冲区中提取数据；

(2) 对数据进行分析、处理；

(3) 将内存单元归还内存池。

IOCP 工作者线程的作用是处理底层数据的流动，其处理数据的时间是非常宝贵的，因此，对数据进行进一步的处理工作需要一个额外的线程单独处理。为了安全地对数据进行处理，需要将所有数据放到一个公共缓冲区中。又因为共有多个 IOCP 工作者线程向公共缓冲区搬移数据，所以，不论是从 Socket 缓冲区向公共缓冲区搬移数据，还是将公共缓冲区的数据搬移至数据处理缓冲区，都需要使用线程同步技术，对公共缓冲区进行加锁、解锁的操作。图 4 中的步骤⑤⑥是在数据处理线程中完成的。

4.5　其它问题讨论

4.5.1　IOCP 连接的建立

本设计中，在需要客户机进行连接操作时，使用 connectEx() 函数投递连接请求。在 IOCP 中，有一个投递的概念，该概念是从重叠 IO 模型开始引入 WinSock 网络编程中的。投递使用的方法是使用重叠结构体，在 WinSock2.2 的版本中，结构体叫做 WSAOVERLAPPED，通过重叠 IO 机制，允许发起一个操作，或者说，发起一次投递，然后在操作完成之后接收到消息。主线程中的 connectEx() 函数就是一种连接投递操作。connectEx() 属于 Winsock2.2 版本的函数，较之 Winsock1.1 的 connect() 函数，connect() 是同步函数，采用阻塞模式建立连接，而 connectEx() 是异步函数，相当于 IO 操作中的投递请求，不会阻塞线程中的执行。

与通常的客户机初始化不同的是，在 IOCP 的客户机中，创建 Socket 之后，需要与本地的一个端口进行绑定，然后再与 IOCP 进行关联，因为 IOCP 处理的 Socket 需要有明确的端口号与之相对应。

需要注意的是 connectEx() 函数具有高性能要求，之前任何在 Socket 上所做的设定属性不会自动复制到已建立连接的 Socket 上。因此，应用程序必须在建立连接之后，调用 setsockopt() 函数在该 Socket 上设置 SO_UPDATE_CONNECT_CONTEXT 属性。这样，建立连接之前的设置才可以传递给此 Socket。

4.5.2　IOCP 工作者线程数量

理论上，为保证工作效率，IOCP 工作者线程的数量根据

CPU 的数量而定，多数情况下线程数量为 CPU 数量的 2 倍加 2，这是出于在线程切换之间的环境切换(context switching)和能够满足于实时处理之间的一种折中。在实际应用中，根据实际连接数和接收大数据量信息的要求，应该改变线程个数以提高数据吞吐能力。在本文第 5 节的测试中，针对 IOCP 线程数和最大并发线程数的不同选取做了相应的测试。

5 测试

本文核心工作在于客户机的 IOCP 实现和高效缓冲区的设置。因此，测试主要针对这两方面进行。

测试环境为：Intel Core(Tm) 2CPU 6300 1.86GHz，内存 2GB，操作系统 Windows XP Professional SP3，100Mbps 局域网。

5.1 连接成功率

测试分为 2 组：第 1 组，验证同步连接与异步连接性能的差距。在 IOCP 线程数相同，连接采用同步连接 connect()和异步连接 connectEx()下的连接进行测试；第 2 组，验证异步连接下 IOCP 线程数目的改变对性能的影响。在测试 Server 端相同，IOCP 线程数不同的情况下，使用 connectEx()进行连接测试。

表 1 显示，在 IOCP 线程数都为 6 的情况下，connectEx()机制比 connect()机制连接成功率高 2%，连接时间降低了一个数量级，减少 93.88%，性能得到大幅提高。

表 1 连接测试

	IOCP 线程数	连接数	成功数	用时(ms)
connectEx()	4	200	199	131
connectEx()	6	200	200	125
connectEx()	8	200	200	131
connectEx()	10	200	199	135
connect()	6	200	196	2043

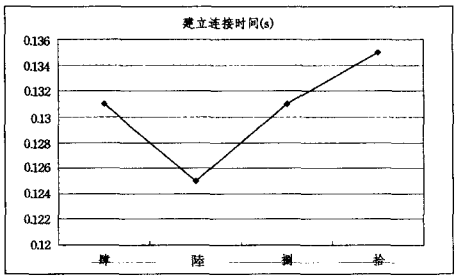

图 5 connectEx()连接测试曲线

图 5 是对表 1 中 connectEx()方式下，不同 IOCP 线程数测试连接的连接时间所绘制的曲线图。从图中可以看出，IOCP 线程数为 $2n+2$(CPU 个数为 n)时，连接成功率最大，用时最少。

5.2 大数据量测试

采用 Visual Studio 2008 开发实现的客户机程序，在 30 个服务器连接的情况下，同时向客户机发送 1.25M 的数据。采用 3 种缓冲方案进行比对测试。

方案一 固定大小内存块缓冲。

方案二 循环队列缓冲。

方案三 内存池＋高效缓冲方案(本文的设计方案)。

上述三种方案的 IO 模型均采用 IOCP 模型，方案的不同之处在于数据缓冲方案的设计上。表 2 中的时间表示从接收到第一包数据至所有应用层数据组包完成所用的时间，表 2 中的内存使用表示内存使用的峰值，表 2 中的线程个数表示数据缓冲单元所用线程个数，即不包括 IOCP 线程数在内的线程数量。

表 2 大数据量测试

	时间(ms)	内存使用(kB)	线程个数
方案一	3640	78,375	30
方案二	3528	78,142	30
方案三	3365	58,157	1

从表 2 中可以看出，方案三所用时间比方案一减少 7.55%，比方案二减少 4.62%，时间的减少主要是由于减少了数据复制次数；方案三的内存使用是方案一的 72.20%，是方案二的 74.42%，内存使用的减少主要是由于内存池的使用；方案三的线程个数是 1，与连接数量无关，而方案二与方案三的线程个数与连接个数相等，当连接数量激增时，此 2 种方案的线程数会相应增加。

测试结果表明，内存池＋高效缓冲方案在接收较大数据量的情况下，资源占用少，缓冲效率高，性能上比传统方式得到显著提高。

5.3 吞吐能力测试

采用 Visual Studio 2008 开发实现的测试客户机程序，在 200 个服务器的连接情况下，分别对 IOCP 工作者线程数进行比较测试。该测试的测试时间从接收到第一包数据开始至应用层数据提取完毕截止。

从表 3 可以看出，在 IOCP 线程数固定的情况下，最大并发线程数为 4 的吞吐率总是高于最大线程数为 2 的情况。在 IOCP 线程数为 6，最大并发线程数为 4 时，吞吐率达到峰值。

表 3 吞吐能力测试(Bytes/s)

IOCP线程数 \ 最大并发线程数	2	4	6	8	10	12	16
4	3972616	4178101	—	—	—	—	—
6	3676388	5211057	3386452	—	—	—	—
8	3655531	4398814	—	3770865	—	—	—
10	3482724	4606698	—	—	2764579	—	—
12	3226485	3676945	—	—	—	2770642	—
16	3716852	3966986	—	—	—	—	3530891

测试结果表明，采用高效缓冲方法其吞吐能力基本满足要求，并且当 IOCP 工作者线程数是 $2n+2$，最大并发线程数是 $2n$ 时(CPU 个数为 n)，吞吐量最大。

结束语 本文针对 IOCP 的客户机应用给出了实现方案，采用异步连接机制增强了连接的效率，并且针对不定长的数据接收采用内存管理模式给出了一种通用性比较强的高效缓冲方案。通过测试，该方案可以满足之前提出的要求。本设计特别适用于连接数量适中，一次接收数据量比较大，需要分包接收的情况。

(下转第 405 页)

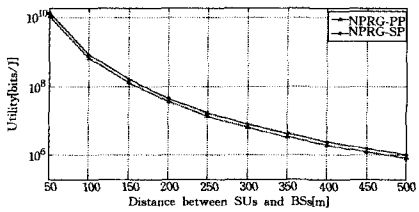

图 5　均衡点效用函数对比

图 3 显示比较了达到均衡点时的两种算法收敛的功率效果，可见 NPRG-SP 算法明显优于 NPRG-PP 算法，也更适用于认知网络环境；另外，图 4 示出两种算法关于 SINR 的比较，可以看出在 150m 以内的用户相比，NPRG-SP 有着更好的 QOS 保证。图 5 中比较了均衡的效用函数，在所有的仿真距离中，NPRG-SP 的效用函数都大于 NPRG-PP 算法。

结束语　我们模拟了一个认知无线电环境，并且将其用户间的关系考虑为非合作的。提出了一个联合功率和速率的算法。通过数值仿真和对比，我们可以发现所提出的算法具有较高的效用值，且在收敛点具有较小的收敛功率和较高的传输速率，达到了保护授权用户的同时满足次级用户 QoS 的目的。

参 考 文 献

[1] Federal Communication Commission. Spectrum Policy Task Force: Report of the spectrum efficiency working group[R]. USA: Spectrum Efficiency Working Group Report

[2] Mitola J, Maguire G. Cognitive radio: making software radios more personal[J]. IEEE Personal Communications, 1999, 6(4): 13-18

[3] Mitola J. Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio [D]. Stockholm: Royal Institute of Technology(KTH), 2000

[4] Jondral F K. Cognitive Radio: A Communications Engineering View [J]. IEEE Wireless Communications, 2007, 14(4): 28-33

[5] Xi Zhang, Su Hang. Opportunistic Spectrum Sharing Schemes for CDMA-Based Uplink MAC in Cognitive Radio Networks [J]. IEEE Journal on Selected Areas in Communications, 2011, 29(4): 715-730

[6] Yates R D, Huang C-Y. Integrated power control and base station assignment[J]. IEEE Trans. Technol., 1995, 12(3): 638-644

[7] Yates R D. A framework for uplink power control in cellular radio systems[J]. IEEE J. Sel. Areas Commun., 1995, 13(7): 1341-1347

[8] Saraydar C U, Mandayam N B, Goodman D J. Efficient power control via pricing in wireless data networks[J]. IEEE Trans. Commun., 2002, 50(2): 291-303

[9] Alpcan T, Basar T, Srikant R, et al. CDMA uplink power control as a noncooperative game[Z]. Wireless Netw., 2002: 659-670

[10] Zhao Wen-tao, Mil u. Distributed Rate and Power Control for CDMA Uplink [J]. IEEE Transactions on Communications, 2004, 14(9): 14-23

[11] Hayajneh M, Abdallah C T. Distributed Joint Rate and Power Control Game-Theoretic Algorithms for Wireless Data[J]. IEEE Communications Letters, 2004, 8(8): 511-513

[12] Musku M R, Chronopoulos A T. A Game-Theoretic Approach to Joint Rate and Power Control for Uplink CDMA Communications[J]. IEEE Transactions on communications, 2010, 58(4): 923-931

[13] Al-Daoud A, Alanyali M, Starobinski D. Secondary pricing of spectrum in cellular CDMA networks[C]// Proc. IEEE DySPAN. Nov. 2007: 535-542

[14] Bagayoko A, Fijalkow I. Power Control of Spectrum-Sharing in Fading Environment With Partial Channel State Information [J]. IEEE Transactions on Signal Processing, 2011, 5(59): 2244-2257

[15] Manosha K B S, Rajatheva N. Joint Power and Rate Control for Spectrum Underlay in Cognitive Radio Networks with a Novel Pricing Scheme[Z]. IEEE, 2010

[16] Zhou P, Yuan W, Liu W, et al. Joint power and rate control in cognitive radio networks: A game-theoretical approach[C]// Communications, 2008. ICC '08. IEEE International Conference. Beijing, May 2008: 3296-3301

(上接第 388 页)

参 考 文 献

[1] Jones A, Ohlund J. Windows 网络编程(第二版)[M]. 杨庆合, 译. 北京：清华大学出版社, 2002: 101-134

[2] 吴星，黄爱萍. 用完成端口实现可扩展的服务器应用[J]. 计算机科学, 2002, 29(11): 144-145

[3] 张静华，张玉明. IOCP 研究及在大规模网络通信系统中的应用[J]. 计算机与现代化, 2004(9): 41-43

[4] 王文武，赵卫东，王志成，等. 高性能服务器底层网络通信模块的设计方法[J]. 计算机工程, 2009, 35(3): 103-105

[5] Kim G-B. A Method of Generating Massive Virtual Clients and Model-based Performance T-est: Quality Software 2005. (QSIC 2005)[C]// Fif-th International Conference. 2005: 250-254

[6] Wang Hui-jiao, Yu Xiao-yong. Research and Impl-ementation of an Intelligent Traffic Monitori-ng System based on IOCP Mechanism: Computer A-pplication and System Modeling(ICCASM)[C]// International Conference on. 2010: V12-656 - V12-659

[7] Zong Xiao-ping, Zhang Yan. Design and Impleme-ntation of the Remote Monitor Server Based on Video Image: Intelligent Networks and Intellig-ent Systems(ICINIS)[C]// 2010 3rd International Conference. 2010: 32-35

[8] 李凌. Winsock2 网络编程实用教程[M]. 北京：清华大学出版社, 2003: 221-225

[9] 苏羽，王媛媛. Visual C++网络游戏建模与实现[M]. 北京：北京科海电子出版社, 2003: 51-52

[10] 陈和平，周静宁，顾晋广，等. IOCP 机制与网络代理服务器实现方法[J]. 计算机应用, 2003, 4(23): 109-110

[11] 侯捷. 池内春秋——Memory Pool 的设计哲学和无痛设计[J]. 程序员, 2002(9): 94-97