**软件详细设计文档**

**ver 1.0**

# 范围和内容

遵守软件设计方案，满足需求的客户机、服务器的软件详细设计文档。

# 2 术语与约定

1. 服务器：项目中接受连接并回弹数据的软件实体。在本项目中，即使使用多线程技术实现，也只相当于一台服务器。
2. 客户机：项目中发起连接并主动发送数据、接收服务器回弹数据的软件实体。在本项目中，若使用多线程技术，则客户机数等于软件开启的用于发送、接收数据的线程数。

# 3 功能模块设计概述

**3.1 基本需求**

1. 客户机能尽自己最大努力向服务器发送数据
2. 服务器能将所有的客户数据全部无损回弹
3. 客户机能统计流量和延时
4. 能对比多线程和基于select框架
5. 服务器具有多线程和各线程基于select的结构，获得更大的服务量和更高的性能。
6. 多个客户机（或多线程设计）向服务器发起大流量“攻击”

**3.2 功能概述**

1. 客户机能尽自己最大努力向服务器发送数据，使用多线程，每个线程中使用一个套接字全力向服务器发送大量简单重复的字符串。
2. 服务器将所有客户数据全部无损回弹，使用多种不同的处理方式进行相应。
   1. 采用select方式，轮询所有套接字。
   2. 采用多线程，每个线程中单套接字，并发处理各个套接字。
   3. 采用多线程与select结合的方式。
3. 客户机能统计流量和延时。
   1. 流量，通过每个线程完成后，返回发送或接收的字符总数，来计算流量。
   2. 延时，则通过在每一帧信息中加入时间戳，在客户端接收到响应之后，通过帧中时间戳计算出延时。

**3.3 主要技术指标**

1. 平均首次发送-末次发送时间
2. 平均首次发送-首次接收时间
3. 平均首次发送-末次接收时间

**3.4 关键问题分析及解决思路**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 问题 | 回答 | 批注 | 更正 |
| 客户端 | | |  |  |
| 1.0 | 客户端如何有效对服务器施压？ | 一个客户端，可选择开启多线程，发送大量重复数据，并接收服务器回复信息。 |  |  |
| 1.1 | 发送什么样的数据？ | 1、大型文件， |  |  |
| 2、重复的字符。 |  |  |
| 1.1.1 | 选择重复字符 | 考虑到本项目的可调的目的，选择重复的字符。 |  |  |
| 1.1.2 | 如何检验数据在传输过程中是否完整？ | 检查总发送和最终接收的数据总和是否相等 |  |  |
| 1.2 | 如何开启线程？ | 使用CreateThread function： **HANDLE WINAPI CreateThread( \_In\_opt\_  LPSECURITY\_ATTRIBUTES  lpThreadAttributes,  \_In\_      SIZE\_T                 dwStackSize,  \_In\_      LPTHREAD\_START\_ROUTINE lpStartAddress,  \_In\_opt\_  LPVOID                 lpParameter,  \_In\_      DWORD                  dwCreationFlags,  \_Out\_opt\_ LPDWORD                lpThreadId );** 具体各参数功能参考MSDN相关文档 |  |  |
| 1.3 | 如何实现延时计算？ | 在每一个线程建立连接后，开始发送前，记录下时间戳。在收到第一次回复后，再记录下时间戳。比较计算延时。 |  |  |
| 在发送的每一帧上加上一定字节记录客户端发送时间。在客户端收到回复后，记录时间。比较计算延时 |  |  |
| 服务器 | | |  |  |
| 2.0 | 服务器采用何种方式应对客户端的大量数据传输请求？ | 1、采用多线程，每个线程管理一个套接字 |  |  |
| 2、采用select轮询管理多个套接字 |  |  |
| 3、采用多线程，每个线程内使用select管理多个套接字 |  |  |
| 2.1 | 如何将套接字与线程一一关联？ | 在每次新建线程时，将套接字作为参数传递给线程所要执行的函数。通过IpParameter。 |  |  |
| 2.2 | 如何处理套接字收与发的关系？ | 1、先发后收 |  |  |
| 2、先收后发 |  |  |
| 2.2.1 | 先发后收可能存在的问题？ |  |  |  |
| 2.2.2 | 先收后发可能存在的问题？ |  |  |  |
| 2.3 | 如何分配每个线程中的套接字数量？ | 1、按照指定的数量，在接收到一定数量的连接之后，创建新线程，将一定量的套接字作为参数传入。 |  |  |
| 2、当主套接字接收了一定量的连接之后，停止在当前线程下的监听，并新建线程，将主套接字作为参数传入，然后在新线程中继续监听接收连接。 |  |  |
| 2.3.1 | 该方案可能存在的问题？ | 该项目中，只要每个线程中套接字个数，与客户端开启的连接个数恰当，并不会出现问题。 而当实际中，如果新建套接字数量未达到指定数量时，并不会新建线程，处理各个套接字上的传输服务。会出现缺少一个新连接而导致的多个连接仅处于初始化的停滞状态。 |  |  |
| 2.3.2 | 该方案可能存在的问题？ |  |  |  |

**3.5 软件模块划分与层次结构关系**

客户机：根据控制流程，分为4个模块：参数输入->线程建立->通信->统计

服务器：根据控制流程，分为3个模块：Selcet框架维护、发送及其异常处理、接收及其异常处理

**3.6 工作原理**

**1) 线程：**

线程，有时被称为轻量级进程(Lightweight Process，LWP），是程序执行流的最小单元。一个标准的线程由线程ID，当前指令[指针](http://baike.baidu.com/view/159417.htm" \t "_blank)(PC），[寄存器](http://baike.baidu.com/view/6159.htm" \t "_blank)集合和[堆栈](http://baike.baidu.com/view/93201.htm)组成。另外，线程是进程中的一个实体，是被系统独立调度和分派的基本单位，线程自己不拥有系统资源，只拥有一点儿在运行中必不可少的资源，但它可与同属一个进程的其它线程共享进程所拥有的全部资源。一个线程可以创建和撤消另一个线程，同一进程中的多个线程之间可以并发执行。由于线程之间的相互制约，致使线程在运行中呈现出间断性。线程也有[就绪](http://baike.baidu.com/view/654230.htm)、[阻塞](http://baike.baidu.com/view/497285.htm)和[运行](http://baike.baidu.com/view/1026025.htm)三种基本状态。就绪状态是指线程具备运行的所有条件，逻辑上可以运行，在等待处理机；运行状态是指线程占有处理机正在运行；阻塞状态是指线程在等待一个事件（如某个信号量），逻辑上不可执行。每一个程序都至少有一个线程，若程序只有一个线程，那就是程序本身。

线程是程序中一个单一的顺序控制流程。进程内一个相对独立的、可调度的执行单元，是系统独立调度和分派CPU的基本单位指[运行](http://baike.baidu.com/view/1026025.htm" \t "_blank)中的程序的调度单位。在单个程序中同时运行多个线程完成不同的工作，称为[多线程](http://baike.baidu.com/view/65706.htm)。

**2) select架构**

select 用于查询设备的状态，以便用户程序获知是否能对设备进行非阻塞的访问，需要设备驱动程序中的poll 函数支持。 驱动程序中 poll 函数中最主要用到的一个 API 是 poll\_wait，其原型如下：

void poll\_wait(struct file \*filp, wait\_queue\_heat\_t \*queue, poll\_table \* wait);

poll\_wait 函数所做的工作是把当前进程添加到 wait 参数指定的等待队列（poll\_table）中。

需要说明的是，poll\_wait 函数并不阻塞，程序中 poll\_wait(filp, &outq, wait)这句话的意思并不是说一直等待 outq 信号量可获得，真正的阻塞动作是上层的 select/poll 函数中完成的。select/poll 会在一个循环中对每个需要监听的设备调用它们自己的 poll 支持函数以使得当前进程被加入各个设备的等待列表。若当前没有任何被监听的设备就绪，则内核进行调度（调用 schedule）让出 cpu 进入阻塞状态，schedule 返回时将再次循环检测是否有操作可以进行，如此反复；否则，若有任意一个设备就绪，select/poll 都立即返回。

select函数的参数将告诉内核：

(1) 我们所关心的描述符。

(2) 对于每个描述符我们所关心的条件（是否读一个给定的描述符？是否想写一个给定的

描述符？是否关心一个描述符的异常条件？）。

(3) 希望等待多长时间（可以永远等待，等待一个固定量时间，或完全不等待）

select从内核返回后内核会告诉我们：

(1) 已准备好的描述符的数量。

(2) 哪一个描述符已准备好读、写或异常条件。

**3.7 各子模块方案设计**

客户机：

参数输入：包括服务器IP、端口、测试选项与参数等。

线程建立：统一完成建立，已建立的线程暂时挂起，建立完成后统一唤醒并分别与服务器通信。一条连接的发送和接收分不同的线程进行。

通信：包括发送、接收及其异常处理。

统计：统计内容包括平均首次发送-末次发送时间，平均首次发送-首次接收时间，平均首次发送-末次接收时间。

服务器：

Selcet框架维护：采用经典seclect框架消息处理设计。

发送及其异常处理：注重对出现的WOULDBLOCK和成功字节数未达到预期的情况妥善处理。

接收及其异常处理：同上。

# 4 存在的问题

1. 是否需要PDU设计需要考虑
2. 客户机每次发送的流字节数和内容需要考虑

# 5 结语

# 本文档是对前面问题列表、软件设计方案、软件实现方案的综合分析，对设计实现中的细节予以重视，并留意一些潜在的问题。