信息网络与协议

实验指导书

（2023 秋季学期）

课 程 教 师

卢 汉 成　　hclu@ustc.edu.cn

助　教

李 伦 升　　**lunsheng23@mail.ustc.edu.cn**

龙 万 青　　**madoren@mail.ustc.edu.cn**

孟　　琪　　**mengqi7788@mail.ustc.edu.cn**

陈 雨 昂　　**yuangchen21@mail.ustc.edu.cn**

注意事项

1) 本课程一共包含 4 个实验，分五周完成。
2) 实验报告通过邮箱进行提交，具体提交方式见实验准备（4）。报告提交截止时间为每周三，迟交会有记录。
3) 第四次实验时间为两周，可以提前完成并提交实验报告。
4) 四次实验内容是彼此独立的，单次实验完成之后请务必点击结束实验按钮释放资源。每次实验直接启动建立好的实验，不需要重复创建。
5) 本课程实验基于未来网实验设施平台进行，如实验过程中遇到问题，请及时通过课程群与助教联系。

实验准备

本次实验基于未来网络实验设施平台 http://ceni.ustc.edu.cn 进行，请通过校园网登录该网站进行实验。关闭网页不会影响实验数据。

一、 登录

在浏览器输入地址 http://ceni.ustc.edu.cn 登录实验平台。

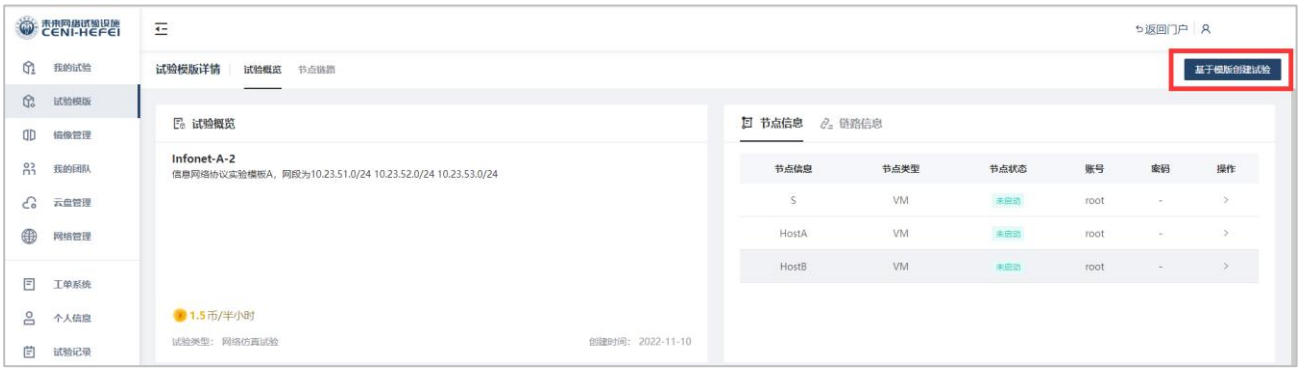
二、 实验创建

由于 ipv4 地址同一网段地址不足够支持所有同学进行实验，现在提供了两套网段，其对应关系如下

主机名	HostA/B	S	S HostA/B
网段 1	10.23.51.0/24	10.23.52.0/24	10.23.53.0/24
网段 2	10.23.54.0/24	10.23.55.0/24	10.23.56.0/24

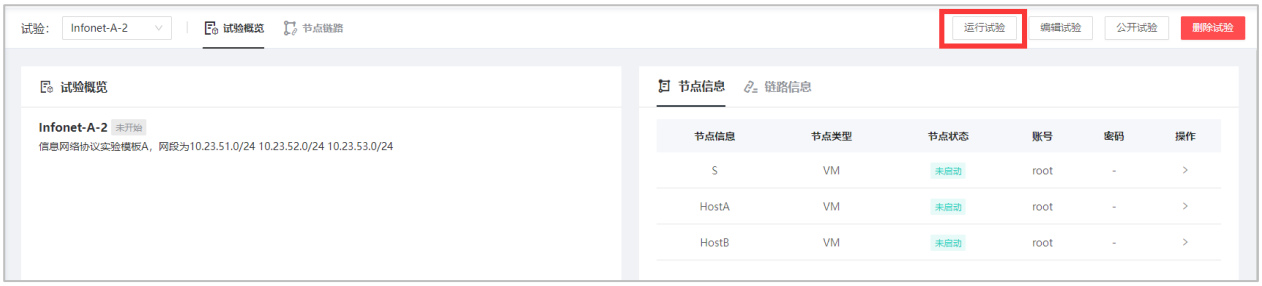
请**学号最后一位数字**为**奇数**的同学选择第一个网段，**偶数**的同学选择第二个网段进行实验。

直接使用实验模板创建实验，网段1实验模板为Infonet-A-2，网段2模板为InfonetB-2。

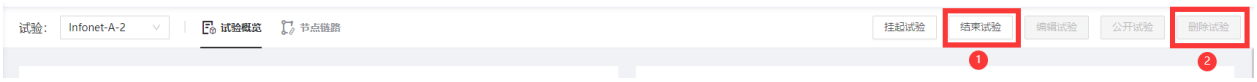


三、 实验运行

点击刚刚新创建的实验，点击运行（此过程可能需要等待一段时间），创建完成之后打开控制台，即可开始实验。请选择以**ustc账号**进行登录，其登录密码为**ustc1958**



注意：四次实验内容是彼此独立的，单次实验完成之后请务必点击**结束实验**按钮释放资源。下次实验直接启动就好，不需要重复创建实验



四、 结果提交

实验报告统一提交至邮箱: ustc_infonet@163.com ， 标题和文件命名规则为：**姓名-学号-信网第x次实验**。前三次实验请提交PDF文件，第四次将报告和代码打包为一个压缩包。

实验三：IPv6 网络编程实验——入门

一、 实验目标

1. 了解 TCP 的 Client/Server 结构的连接建立过程。
2. 掌握网络 Socket 编程的基本概念和基本方法。
3. 掌握 TCP 的 Client/Server 结构的通信的编程。

二、 实验原理

1. TCP 通信

传输层和应用层之间进行的数据交换称为报文（Message），而在传输层和网络层之间进行交换的数据称为数据报（Datagram）。传输层可以使用传输控制协议（TCP）来封装数据。TCP 协议面向连接，使用字节流传送服务，是可靠的。

在面向连接的 Client/Server 结构中：服务器首先启动，通过调用 socket() 建立一个套接口，然后调用 bind() 将该套接口和本地网络地址联系在一起，再调用 listen() 使套接口做好侦听的准备，并规定它的请求队列的长度，之后就调用 accept() 来接收连接。客户在建立套接口后就可调用 connect() 和服务器建立连接。连接一旦建立，客户机和服务器之间就可以通过调用 read() 和 write() 来发送和接收数据。最后，待数据传送结束后，双方调用 close() 关闭套接口。

2. 套接字 socket

套接字（socket）是通信的基石，是支持 TCP/IP 协议的网络通信的基本操作单元。它是网络通信过程中端点的抽象表示，包含进行网络通信必须的五种信息：连接使用的协议，本地主机的 IP 地址，本地进程的协议端口，远地主机的 IP 地址，远地进程的协议端口。

应用层通过传输层进行数据通信时，TCP 会遇到同时为多个应用程序进程提供并发服务的问题。多个 TCP 连接或多个应用程序进程可能需要通过同一个 TCP 协议端口传输数据。为了区别不同的应用程序进程和连接，许多计算机操作系统为应用程序与 TCP ／ IP 协议交互提供了套接字 (Socket) 接口。应用层可以和传输层通过 Socket 接口，区分来自不同应用程序进程或网络连接的通信，实现数据传输的并发服务。

Socket 连接，至少需要一对套接字，分为 clientSocket，serverSocket。连接分为 3 个步骤：

a) 服务器监听：服务器并不定位具体客户端的套接字，而是时刻处于监听状态。

b) 客户端请求：客户端的套接字要描述它要连接的服务器的套接字。提供地址和端口号，然后向服务器套接字提出连接请求。

c) 当服务器套接字收到客户端套接字发来的请求后，就响应客户端套接字的请求，并建立一个新的线程，把服务器端的套接字的描述发给客户端，一旦客户端确认了此描述，就正式建立连接。而服务器套接字继续处于监听状态，继续接收其他客户端套接字的连接请求。

三、 实验内容

1. 阅读本文档中的两个程序 TCPClient.c、TCPServer.c。
2. 编译与运行程序：
   a) 新建文件 TCPClient.c、TCPServer.c，参考文档中的示例编写程序。
   b) 在保存代码文件的文件夹中打开两个终端。
   c) 首先编译运行服务器：
      I. 执行 gcc TCPServer.c -o TCPServer 编译服务器程序。
      II. 执行命令./TCPServer 运行服务器程序。
   d) 其次编译运行客户端：
      I. 执行 gcc TCPClient.c -o TCPClient 编译客户端程序。
      II. 执行命令./TCPClient ::1 运行客户端程序。
      注意 这里的::1 表示回环地址，即这个示例程序完成了同一个主机上两个进程之间的通信。
3. 请根据实验原理和[The GNU C Library Reference Manual]手册理解本实验示例代码的含义。

四、 思考题

1. 上述示例程序是一个基于 C/S 模型的 TCP 通信程序。请根据对示例程序的理解将 socket()、write()、close()、read()、listen()、bind()、connect()、accept() 这八个基本函数填入如图 1 所示的典型 C/S 模型 TCP 通信程序流程图中。

2. 图 2 给出了一张 TCP 状态转移图。请根据你的理解将执行上题中每个函数之后 TCP 连接所处的状态在图中标出，标识的方式是在状态旁写出 [主机 A/主机 B 执行函数名，主机 A/主机 B 处于此状态]。根据这个状态转移图的理解以及相关资料简要说明 SYN Flood 攻击的实现原理。

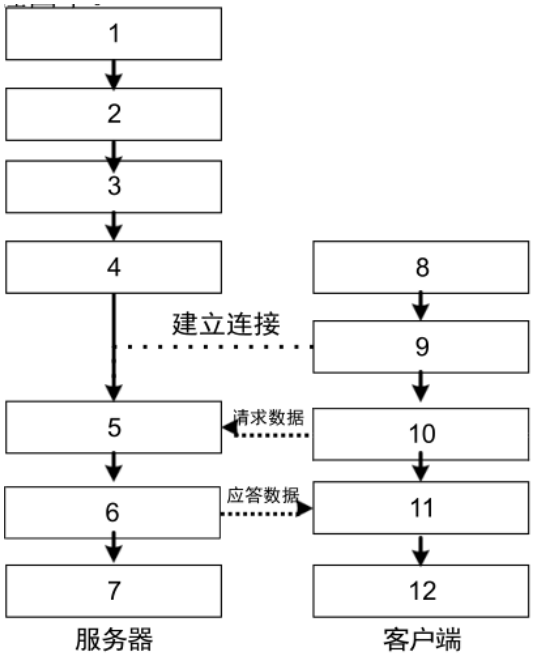


图 1: 基于 C/S 模型的 TCP 通信程序流程图

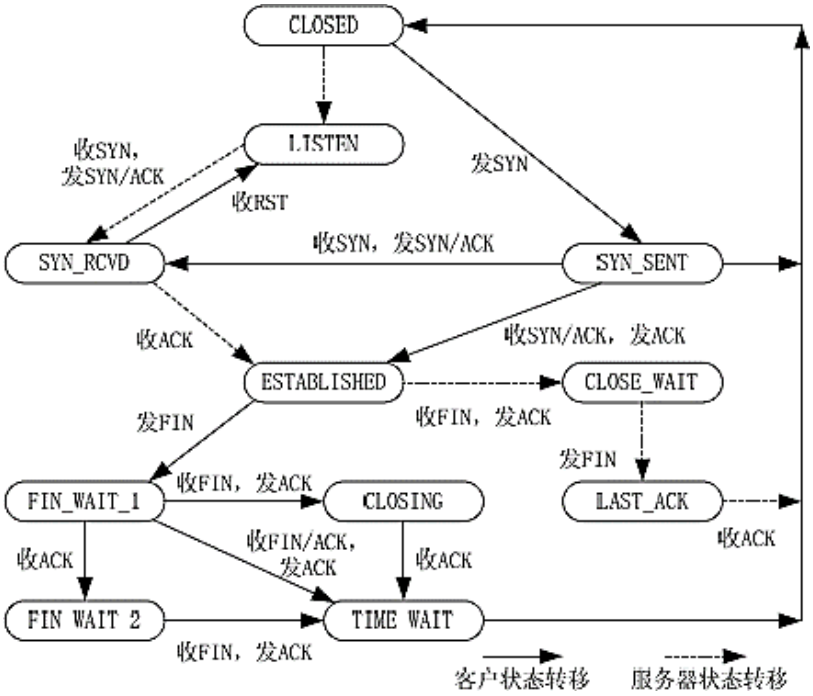


图 2: TCP 状态转移图

五、 实验报告

报告要求：电子版报告，内容包括实验题目、实验原理 (对示例代码中的语句进行解释)、实验结果(示例程序运行截图)、实验结果、实验收获。

注意 实验报告及时提交，晚交会记录并扣分。

实验报告截止时间：11 月 26 日 23 时 59 分 59 秒

附录 A　实验三参考代码

TCPServer.c

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define MAXLINE 1024
#define TRUE1

int main(int argc, char **argv)
{
    int sockfd, fd, n, m;
    char line[MAXLINE + 1];
    struct sockaddr_in6 servaddr, cliaddr;
    time_t t0 = time(NULL);
printf("time #: %ld\n", t0);
fputs(ctime(&t0), stdout);

        if((sockfd = socket(AF_INET6, SOCK_STREAM, 0))
    < 0) perror("socket error");

    bzero(&servaddr,
sizeof(servaddr));
servaddr.sin6_family = AF_INET6;
servaddr.sin6_port = htons(20000);
servaddr.sin6_addr = in6addr_any;

        if(bind(sockfd, (struct sockaddr*)&servaddr, sizeof(servaddr))
    == -1) perror("bind error");

        if(listen(sockfd, 5)
    == -1) perror("listen
    error");

    while(TRUE) {

        printf("> Waiting clients ...\r\n");

    socklen_t clilen = sizeof(struct sockaddr);
        fd = accept(sockfd, (struct sockaddr*)&cliaddr, &clilen);
```

```
            if(fd == -1)
                { perror("ac cept
               error");
               }

               printf("> Accepted.\r\n");

while((n = read(fd, line, MAXLINE)) > 0)
                { line[n] = 0;
                    if(fputs(line, stdout)
                  == EOF) perror("fputs
                  error");
               }
               close(fd);
          }

          if(n < 0) perror("read error");


      }

     exit(0);
```

TCPClient.c

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <time.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define MAXLINE 1024
#define TRUE1

int main(int argc, char **argv)
{
    int sockfd, n, m;
    char line[MAXLINE + 1];
struct sockaddr_in6 servaddr;
time_t t0 = time(NULL);
printf("time #: %ld\n", t0);
fputs(ctime(&t0), stdout);

    if(argc != 2)
        perror("usage: a.out <IPaddress>");

        if((sockfd = socket(AF_INET6, SOCK_STREAM,
    0)) < 0) perror("socket error");

    bzero(&servaddr,
sizeof(servaddr));
servaddr.sin6_family = AF_INET6;
servaddr.sin6_port = htons(20000);

        if(inet_pton (AF_INET6, argv[1], &servaddr.sin6_addr)
    <= 0) perror("inet_pton error");

    if(connect(sockfd, (struct sockaddr*)&servaddr, sizeof(servaddr)) <
0)
```

```
        perror("connect error");

    while(fgets(line, MAXLINE, stdin) != NULL)
        { send(sockfd, line, strlen(line), 0);
    }

    close(sockfd);

    exit(0);
}
```