
实验报告

1. 实验目的

目的：

- (1) 掌握网络应用程序的开发方法
- (2) 掌握 Client/Server 结构软件的设计与开发方法
- (3) 掌握 Socket 机制的工作原理
- (4) 掌握基于 Client/Server 结构的 Windows Socket TCP/UDP 程序设计方法

要求完成：

- (1) 使用集成开发环境编写“Hello World”程序
- (2) 运行 simple-talk 例程，理解代码并观察现象
- (3) 修改 simple-talk，编制 duplex-talk 程序，支持 client 和 server 的双向通信
- (4) 利用 Windows 时间函数，编制简单的定时器，模拟 client 和 server 之间的 stop-and-wait 的动作

正常通话：成功收到 ACK

超时重传：在时限内未能收到 ACK，超时自动重传

2. 实验环境

操作系统：Windows

编程工具和集成开发环境：CodeBlocks

桌号：505

机器 IP 地址：192.168.1.155

3. 实验内容与结果

- (1) 程序整体功能：

本次实验基于 simple-talk 总共编写了三个 Client/Server 结构类型的 Socket 通信模块，三个模块整体功能如下：

- A. 双向通信 (duplex-talk)：服务端开启一个 Socket 侦听相应端口等待连接，若有客户端发起连接，则开启一个新的 Socket 与该客户端进行连接，与客户端连接后能够实现二者之间的双向通信。

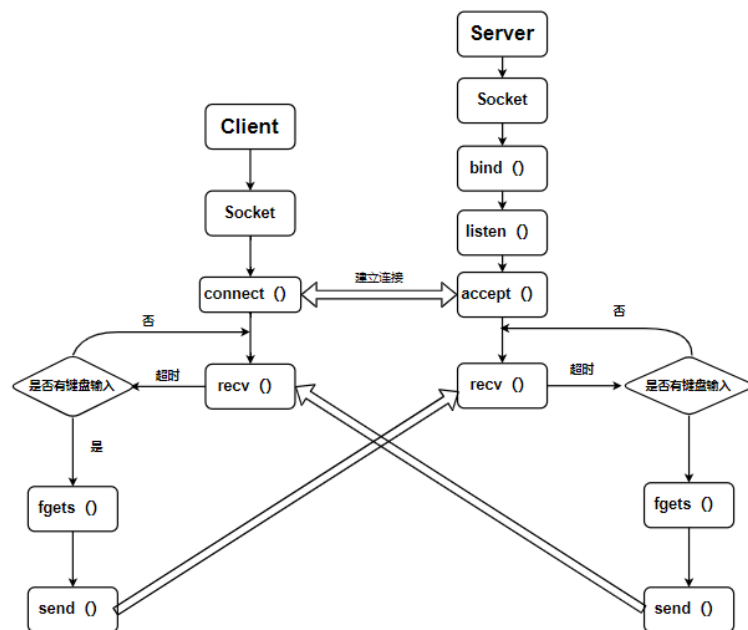


图 1 双向通信流程图

- B. 模拟 stop-and-wait 正常通信（duplex-talk-normal）：基于客户端与服务器之间的双向通信（即模块 A），本模块新增了消息的确认机制，即客户端和服务端接收到对方的信息后，都会回应一个 ACK！表示已收到信息，若客户端给服务器发送空的消息，则表示请求断开与服务端之间的连接。

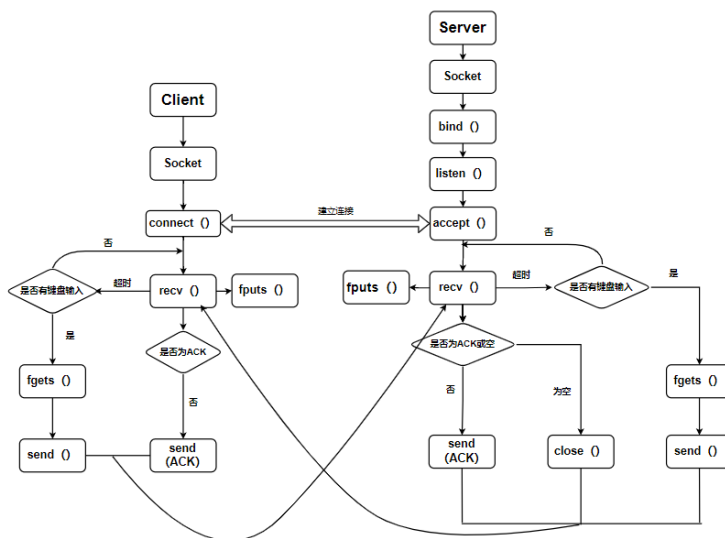


图 2 stop-and-wait 正常通信流程图

- C. 模拟 stop-and-wait 超时重传（duplex-talk-overtime）：基于客户端与服务器之间的双向通信（模块 A），模拟 stop-and-wait 通信机制中发送方未在一定时间内收到 ACK 进而触发超时重传的现象（实际接收方能接收到信息，只是在接收三次后才会向发送方发送 ACK 表明已收

到信息), 发送方直到收到接收方的 ACK 后才会停止重传。

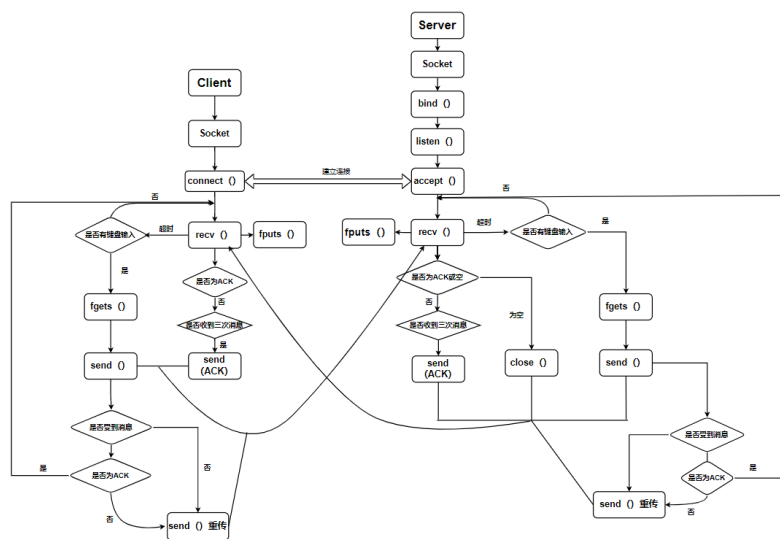


图 3 stop-and-wait 超时重传流程图

(2) 程序组成及各模块/函数功能;

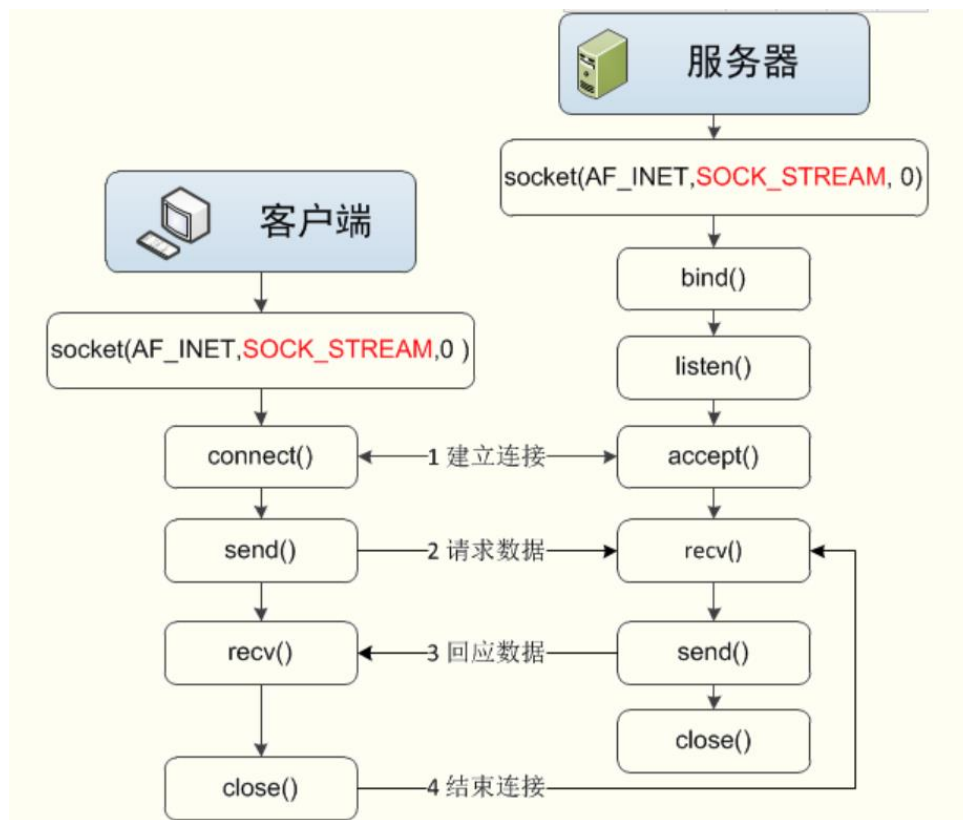


图 4 Socket 通信整体结构框图

A. 建立连接

Server: bind (), listen (), accept ()

服务器创建一个 Socket，并绑定本地的一个端口，然后侦听该端口，若有客户端请求与该端口进行连接，服务器接受后建立新的 Socket 与客户端进行连接。

Client: connect ()

客户端向服务器相应端口请求建立连接

B. 发送/接收消息

Client/Server: send ()、recv ()

在客户端与服务器建立连接后，二者均可看作消息的发送方和接收方。若为模拟 stop-and-wait 通信机制，正常通信时，接收方每收到一条消息（对方回复的 ACK 除外），都会向发送方回复 ACK；超时重传时，发送方发送消息后，会侦听接收方是否回复了 ACK，若在一定时间内未收到 ACK，发送方会对该消息进行重传，而接收方只有在收到三条消息后才会回复 ACK。

C. 断开连接

Client/Server: close ()

在客户端和服务器建立连接后，双方可进行双向通信。若客户端向服务器发送一条空消息，则视为请求断开连接，服务器收到该消息后断开与客户端之间的连接。

(3) 重要的数据结构，模块/函数算法；

A. Winsock 的启动和终止---WSAStartup()和 WSACleanup()

由于 Winsock 服务是以动态链接库的形式实现的，所以在使用前必须调用 WSAStartup 函数对其进行初始化，协商 Winsock 的版本支持，并分配必要的资源。在应用程序关闭套接字连接后，还需要调用 WSACleanup 函数终止对 Winsock 库的使用，并释放资源。

B. 创建套接字---socket()

调用 socket 创建一个流套接字，函数声明如下：

SOCKET socket(int af,int type,int protocol);

参数说明

af: 指定网络地址族，一般为 AF_INET。

type: 指定套接字类型，可选的取值如下：

SOCK_STREAM 流套接字。

SOCK_DGRAM 数据报套接字。

protocol: 指定网络协议，一般为 0，表示默认的 TCP/IP 协议。

C. 绑定本地地址---bind()、侦听连接---listen()

成功创建了 Socket 之后，就应该选定通信的对象。调用 bind()函数可以将本地地址绑定到套接字上。

绑定成功后，调用 `listen` 函数用于设置套接字的等待连接状态。

D. 建立套接字连接---`accept()`和 `connect()`

进入监听状态后，通过调用 `accept` 函数使套接字做好接受客户连接的准备。

客户进程调用 `connect` 函数可以主动提出连接请求。

E. 面向连接的数据传输---`send()`和 `recv()`

客户和服务端应用程序都用 `send` 函数来向 TCP 连接的另一端发送数据。客户程序一般用 `send` 函数向服务器发送请求，而服务器则通常用 `send` 函数来向客户程序发送应答。

```
int send( SOCKET s, const char FAR *buf, int len, int flags );
```

参数说明：

s: 指定发送端套接字描述符；

*buf: 指明一个存放应用程序要发送数据的缓冲区；

len: 指明实际要发送的数据的字节数；

flags: 一般置 0。

客户和服务端应用程序都用 `recv` 函数从 TCP 连接的另一端接收数据。

```
int recv( SOCKET s, char FAR *buf, int len, int flags );
```

参数说明：

s: 指定接收端套接字描述符；

*buf: 指明一个缓冲区，该缓冲区用来存放 `recv` 函数接收到的数据；

len: 指明 buf 的长度；

flags: 一般置 0。

F. 关闭套接字---`closesocket()`

此函数关闭套接字 s，并释放分配给该套接字的资源。如果套接字关联一个 TCP 连接，则该连接同时被释放。

G. Windows 时间函数-----`GetLocalTime()`

读取当前的系统时间，可精确到毫秒级，可以为每次通信建立时间戳。

H. 设置超时重传函数---`setsockopt()`

设置超时等待时间 `Timeout=xxx`；

```
setsockopt(socket, SOL_SOCKET, SO_RCVTIMEO,
```

```
(char *)&Timeout, sizeof(int));
```

一旦 `recv()` 超时 `Timeout`，将返回 -1，此时客户端应自动重发。

I. 判断是否有键盘输入---`kihit()`

用于非阻塞的响应键盘输入事件。

(4) 程序清单（手写或者打印后作为附件）

- duplex-talk
- duplex-talk-normal
- duplex-talk-overtime

每个文件夹中均包含

- client
- server

客户端和服务端文件中均包含相应的程序

- main.c
- main.exe
- main.o

(5) 程序的运行和测试结果（提供截图）

A. 双向通信（duplex-talk）

服务器：

```
C:\WINDOWS\system32\cmd.exe - D:\桌面\计网实验\duplex-talk\server\main.exe
Microsoft Windows [版本 10.0.19044.1766]
(c) Microsoft Corporation。保留所有权利。

C:\Users\15387>D:\桌面\计网实验\duplex-talk\server\main.exe
server is ready in listening ...
received a connection from 127.0.0.1 :
16:22:41 [duplex-talk] client: Hello
16:22:53 [duplex-talk] client: How are you
16:22:59 [duplex-talk] server: fine, thanks, and you?
16:23:27 [duplex-talk] client: fine too
16:24:07 [duplex-talk] client: Good bye
16:24:11 [duplex-talk] server: Bye
```

图 5 双向通信（Server）

客户端

```
C:\WINDOWS\system32\cmd.exe - D:\桌面\计网实验\duplex-talk\client\main.exe localhost
Microsoft Windows [版本 10.0.19044.1766]
(c) Microsoft Corporation。保留所有权利。

C:\Users\15387>D:\桌面\计网实验\duplex-talk\client\main.exe localhost
client is connecting to localhost
16:22:36 [duplex-talk] client: Hello
16:22:46 [duplex-talk] client: How are you
16:23:12 [duplex-talk] server: fine, thanks, and you?
16:23:21 [duplex-talk] client: fine too
16:24:00 [duplex-talk] client: Good bye
16:24:12 [duplex-talk] server: Bye
```

图 6 双向通信（Client）

（二者的消息的时间差主要来源于程序切换显示的延迟，而非实际传输时间）

B. stop-and-wait 正常通信（duplex-talk-normal）

服务器：

```

C:\WINDOWS\system32\cmd.exe - D:\桌面\计网实验\duplex-talk-normal\server\main.exe
C:\Users\15387>D:\桌面\计网实验\duplex-talk-normal\server\main.exe
server is ready in listening ...
received a connection from 127.0.0.1 :
16:30:32 [duplex-talk] client: Hello
16:30:32 [duplex-talk] server: ACK!
16:30:49 [duplex-talk] client: How are you?
16:30:49 [duplex-talk] server: ACK!
16:30:52 [duplex-talk] server: Fine, thanks, and you?
16:31:14 [duplex-talk] client: ACK!
16:31:29 [duplex-talk] client: Fine too
16:31:29 [duplex-talk] server: ACK!
16:31:36 [duplex-talk] client: Good bye
16:31:36 [duplex-talk] server: ACK!
16:31:41 [duplex-talk] server: Bye
16:31:44 [duplex-talk] client: ACK!
16:31:49 [duplex-talk] client:
16:31:49 [duplex-talk]: empty message is received
16:31:49 [duplex-talk] server: ACK!
16:31:49 [duplex-talk]: connection from 127.0.0.1 is terminated
received a connection from 127.0.0.1 :
16:32:03 [duplex-talk] client: Hello
16:32:03 [duplex-talk] server: ACK!
16:32:11 [duplex-talk] client: Bye
16:32:11 [duplex-talk] server: ACK!
16:32:45 [duplex-talk] client:
16:32:45 [duplex-talk]: empty message is received
16:32:45 [duplex-talk] server: ACK!
16:32:45 [duplex-talk]: connection from 127.0.0.1 is terminated

```

图 7 Stop-and-wait 正常通信 (Server)

客户端:

```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.19044.1766]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\15387>D:\桌面\计网实验\duplex-talk-normal\client\main.exe localhost
client is connecting to localhost
16:30:27 [duplex-talk] client: Hello
16:30:32 [duplex-talk] server: ACK!
16:30:42 [duplex-talk] client: How are you?
16:30:49 [duplex-talk] server: ACK!
16:31:14 [duplex-talk] server: Fine, thanks, and you?
16:31:14 [duplex-talk] client: ACK!
16:31:25 [duplex-talk] client: Fine too
16:31:29 [duplex-talk] server: ACK!
16:31:32 [duplex-talk] client: Good bye
16:31:36 [duplex-talk] server: ACK!
16:31:44 [duplex-talk] server: Bye
16:31:44 [duplex-talk] client: ACK!
16:31:49 [duplex-talk] client:
16:31:49 [duplex-talk]: empty message is sent to server
16:31:49 [duplex-talk]: connection is terminated

C:\Users\15387>D:\桌面\计网实验\duplex-talk-normal\client\main.exe localhost
client is connecting to localhost
16:32:00 [duplex-talk] client: Hello
16:32:03 [duplex-talk] server: ACK!
16:32:08 [duplex-talk] client: Bye
16:32:11 [duplex-talk] server: ACK!
16:32:45 [duplex-talk] client:
16:32:45 [duplex-talk]: empty message is sent to server
16:32:45 [duplex-talk]: connection is terminated

```

图 8 Stop-and-wait 正常通信 (Client)

C. stop-and-wait 超时重传 (duplex-talk-overtime)

其中超时时间设置为 1 秒，即 1 秒后未收到 ACK 后进行重传

服务器:


```

C:\WINDOWS\system32\cmd.exe - D:\桌面\计网实验\duplex-talk-overtime\server\main.exe
Microsoft Windows [版本 10.0.19044.1766]
(c) Microsoft Corporation。保留所有权利。

C:\Users\15387>D:\桌面\计网实验\duplex-talk-overtime\server\main.exe
server is ready in listening ...
received a connection from 127.0.0.1 :
16:38:38 [duplex-talk] client: Hello
16:38:38 [duplex-talk] server:ACK!
16:39:10 [duplex-talk] client: How are you?
16:39:10 [duplex-talk] server:ACK!
16:39:17 [duplex-talk] server: Fine,thanks,and you?
16:39:32 [duplex-talk] server(time out): Fine,thanks,and you?
16:39:33 [duplex-talk] server(time out): Fine,thanks,and you?
16:39:33 [duplex-talk] client: ACK!
16:39:47 [duplex-talk] client: Fine too
16:39:47 [duplex-talk] server:ACK!
16:39:57 [duplex-talk] client: Bye
16:39:57 [duplex-talk] server:ACK!
16:39:59 [duplex-talk]: empty message is received
16:39:59 [duplex-talk] server: ACK!
16:39:59 [duplex-talk]: connection from 127.0.0.1 is terminated
    
```

图 9 Stop-and-wait 超时重传 (Server)

客户端:

```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.19044.1766]
(c) Microsoft Corporation。保留所有权利。

C:\Users\15387>D:\桌面\计网实验\duplex-talk-overtime\client\main.exe localhost
client is connecting to localhost
16:38:33 [duplex-talk] client: Hello
16:38:37 [duplex-talk] client(time out): Hello
16:38:38 [duplex-talk] client(time out): Hello
16:38:38 [duplex-talk] server: ACK!
16:39:00 [duplex-talk] client: How are you?
16:39:09 [duplex-talk] client(time out): How are you?
16:39:10 [duplex-talk] client(time out): How are you?
16:39:10 [duplex-talk] server: ACK!
16:39:33 [duplex-talk] server: Fine,thanks,and you?
16:39:33 [duplex-talk] client:ACK!
16:39:41 [duplex-talk] client: Fine too
16:39:46 [duplex-talk] client(time out): Fine too
16:39:47 [duplex-talk] client(time out): Fine too
16:39:47 [duplex-talk] server: ACK!
16:39:53 [duplex-talk] client: Bye
16:39:56 [duplex-talk] client(time out): Bye
16:39:57 [duplex-talk] client(time out): Bye
16:39:57 [duplex-talk] server: ACK!
16:39:59 [duplex-talk] client:
16:39:59 [duplex-talk]: empty message is sent to server
16:39:59 [duplex-talk]: connection is terminated
    
```

图 10 Stop-and-wait 超时重传 (Client)

4. 实验中的问题

(1) fgets、recv 函数阻塞问题

在实现双向通信时,我发现当程序运行至读取键盘输入的 fgets 函数和 Socket 接收函数 recv 函数时,会阻塞在这两个函数里,导致程序无法继续运行,而这两个函数分别是发送和接收所必须经过的步骤,所以无法利用条件语句 if 进行双向通信中发送和接收的判断,进而无法实现双方的任意发送或接收,为双向通信的实现带来了困难。

解决方法:对于这个问题,我想到两种可行的思路,一种是利用多线程,同时完成两个函数的阻塞运行,另一种方法是找其

它函数协助，避免这两个函数陷入阻塞或陷入阻塞后能自动退出。最终我选择了第二种方法，相关辅助语句与函数如下：

setsockopt():该函数能够设置超时机制，若 **recv()**在一定时间内未接收到消息可以自动退出，从而避免了 **recv** 函数的阻塞问题。

kbhit(): 该函数能够判断是否有键盘输入，并且为非阻塞函数，可以利用该函数的返回值作条件判断，选择是否运行 **fgets** 函数，即只有键盘输入时（用户选择发送），才会运行 **fgets** 函数进行读取，从而避免了 **fgets** 函数的阻塞问题。

（2）ACK 消息的循环回复

在实现 **stop-and-wait** 机制时，发送方未对接收的信息进行判别，其接收到接收方回复的 **ACK** 后，也会回复一个 **ACK**，最终导致收发双方循环发送 **ACK**。

解决方法：接收方对每次接收到的信息进行检验，若为 **ACK**，则不进行 **ACK** 确认回复；若为空消息，则断开连接；其它消息一律回应 **ACK** 表明确认接收。

（3）超时重传中重传的消息无法发送

在模拟 **stop-and-wait** 中超时重传机制时，发送方每发送一条消息，即会侦听判定接收方是否回复 **ACK**，若一定时间内未回复，则选择重传，然而重传的消息发送后接收方无响应。

解决方法：将判断条件改为是否收到信息，若一定时间内未收到，则重传消息，若收到消息，判断是否为 **ACK**，若不是 **ACK**，仍然选择重传，问题得以解决。（但之前问题的具体原因仍然未知）

附件

1. 程序源代码

见代码附件

2. 参考文献列表

[1] Windows_Socket_编程.pdf

[2] 实验指导书_7.1 【Windows Socket 编程】.pdf

[3] Peterson,L.L. BruceS.Davie 计算机网络 系统方法 机械工业出版社