高级网络编程

课程设计

名　　称：基于多播的聊天室

日　　期：2019年6月23日

姓　　名：王梓岩

学　　号：71Y16118

1 需求分析

在unix环境下搭建一个聊天室，用户可在聊天室中发送文本和文件。凡是在多播组的主机，均可接收到该文本或文件。当有新用户加入或老用户退出多播组时，其余主机也可被通知到。

2 开发环境

Linux ubuntu 4.13.0-39-generic X64 （共计三台机器，均为该操作系统）

3 设计思路

多播，也叫组播，将局域网中同一业务类型主机进行了逻辑上的分组，进行数据收发的时候其数据仅仅在同一分组中进行，其他的主机没有加入此分组不能收发对应的数据。

ICMP回应请求（echo-request）和应答消息（echo-reply）用于诊断两个系统（主机或路由器）之间是否多播的地址是特定的，D类地址用于多播。D类IP地址就是多播IP地址，即 224.0.0.0 至 239.255.255.255 之间的IP地址，并被划分为三类：局部多播地址：在 224.0.0.0 至 224.0.0.255 之间，这是为路由协议和其他用途保留的地址，路由器并不转发属于此范围的IP包；预留多播地址：在 224.0.1.0 至 238.255.255.255 之间，可用于全球范围（如Internet）或网络协议；管理权限多播地址：在 239.0.0.0 至 239.255.255.255 之间，可供组织内部使用，类似于私有IP地址，不能用于Internet，可限制多播范围。本次实验我们选用 224.0.0.111 作为多播地址。

首先，发送和接收必须要同时进行，而两个操作均属于卡死式的行为，因此必须要建立两个线（进）程分别负责。通讯的流程较为简单，初始化阶段需要建立用于多播的套接口，之后设置多播的参数，例如超时时间TTL，本地回环许可LOOP等。以上参数均初始化后就可以将主机加入多播组，之后由发送方发送一个消息，其他接收方将都会接收到该信息，并将其打印在屏幕上或写入文件。通信结束后主机离开多播组。

考虑到当今主流的通讯软件（如QQ）文件与文本的传输是分离的，即发（收）文件的同时也可同时发（收）文本，因此文件与文本也要用不用的线程来处理，算上收发各自需要一个线程，本程序共计有4个线程并发。由于线程间通信花费较高，

因此本程序直接将两者端口分离，文本由7777号端口接收，文件由7778号端口接收，省去了每次解析报文头部的麻烦。

4 相关模块

4.1 主函数

主函数的逻辑非常清晰：首先进行初始化，建立发送、接收文本和文件的四个套接口sendSock，recvSock， sendFileSock和recvFileSock；建立多播地址和本地地址；建立多播请求；获取本机在局域网的IP地址（详细方法见下）；以及等待用户输入自己的用户名。

之后调用两次fork()方法，将主进程分为3个进程，分别处理发文本、收文本和收文件，处理发文件的线程的建立会在后面提到，目前建立的三个线程均为卡住式线程，需要等待用户输入或从IO流中读取信息。

4.2 接收文本

方法recvText()用于接收文本，该模块由fork()方法分出的一个线程单独执行。必须指出的是，fork()方法会将原有的进程完全复制一份，也就是说两个线程对于同名变量不享有共同的地址。所以此时之前建立所有的套接口都各有2个，由于该进程只负责接收文本，因此可将其他无关套接口关闭，只保留用于收文本的套接口recvSock，其他进程也会做类似的处理，以后将不再阐述。

接收文本的所有操作均置于一个死循环中，表示持续接收。调用recvfrom()方法，从指定端口读取IO流的信息，之后将其打印在屏幕上即可。

4.3 发送文本

方法sendText()用于发送文本，同样地，该方法由某一线程单独执行。由于在需求阶段指出，当有一新用户加入聊天后，组内所有成员均可知道他的到来，因此会首先调用sendto()方法发送一个内容为“用户名（IP地址） has joined.”的消息。

之后进入死循环，循环内的逻辑为调用fgets()方法等待用户从键盘输入，之后通过sendto()方法将消息发送给多播组。但是有一点必须要注意，本程序为控制台程序，要发送文件就必须通过规定原语来实现，现规定文件的发送语句为：“file://

+ 文件路径”，如“file:///home/aaa.txt”就表示要发送根目录下的home目录中的aaa.txt文件。因此，对于用户的每一个输入，必须先调用strcmp()方法判定前缀是否为“file://”，如果不是，说明这是普通的聊天信息，将其发送；如果是，则会触发fork()方法分出专门负责发文件的子进程，此时父进程直接等待下一次用户输入，子进程获得文件路径，即输入的字符串除去前缀的部分，之后调用sendFile()方法发送文件。

4.4 发送文件

方法sendFile()用于发送文件，该方法接受一个字符串类型的参数，为文件路径。规定文件的发送方式为第一个报文的前8位是文件大小（long类型），之后是文件名。后面的报文为文件内容。

首先利用结构体stat获取文件大小，此时也会判定文件路径是否合法，不合法直接退出。unix操作系统下推荐使用stat来获取文件大小，此方法无需将文件装载进内存，因此效率较高。由于规定了文件大小占8个字节，因此直接将这个数字的16进制写入报文。文件名称可调用strrchr()方法，该方法可以返回文件路径字符串中最后一个“/”位置的指针，该指针以后的部分即为文件名。

填写好第一个报文后，调用fopen()方法读取文件，之后按二进制方式读取文件，缓冲区的大小为2048，即每次最多读取2KB大小。调用fread()方法将读出的内容写入报文并发送。文件读取完毕后，关闭文件流和套接口，调用exit()方法结束该进程。可以发现发送文件的线程是按需请求的，这是因为考虑到发文件的频率不如发文本，没有必要将该线程一直在后台挂起。

4.5 接收文件

方法recvFile()用于接收文件，该方法由某一线程单独执行。与接收文本的逻辑类似，所有操作均位于死循环内。当收到报文时，首先会解析出文件大小和文件名，调用fopen()方法新建文件，为了避免重名，新建的文件一律以“接收时间 + 原文件名”的方式命名。之后收到的报文是文件内容，调用fwrite()方法将其写入文件，接收完毕后关闭该文件，等待接收下一个文件。

4.6 退出聊天

前面需求指出，我们希望当有人退出聊天时，其他组内成员知道他已退出，该功能可以整合进正常关闭程序的流程中。由于所有的进程都位于死循环中，因此

必须采用其他方法中断这些循环。这里没有采用规定退出程序的原语，而是直接借助传统的unix退出程序的快捷键：Ctrl + C，unix为我们提供了捕获这一信号的系统调用：方法signal()。该方法有2个参数，第一个参数是信号种类，这里我们填SIGINT，第二个参数是触发的函数指针，这个函数用于进行后续操作。在本程序中，方法exitChat()作为处理退出聊天的模块。该方法主要进行两个操作：发送一个自己离开的消息给多播组，关闭所有活跃的套接口，最后调用exit()方法正常结束程序。

4.7 获取本机IP

考虑到虽然用户可以在进入聊天室前输入自己的昵称，但仍不能排除有重名的情况，因此会对每一条信息同时标注用户昵称和IP地址。unix操作系统下获取本机IP的方法为借助ioctl函数获取主机的全部网络接口信息。ioctl函数可以获取所在主机的全部网络接口信息，包括接口地址、是否支持广播等。遍历所有网卡信息，排除名称为lo的本地回环网卡后，就得到了当前网卡的全部信息，记录下对应的IP地址即可。

5 实验结果分析及结论

总计要使用三台主机，IP分别为 192.168.188.128 ，192.168.188.129和192.168.188.130 。以下用三台机器的用户名aaa、bbb和ccc代指，三台主机的控制台背景色分别为紫、白、黑，以便于区分。但是，现在的视频文件往往较大，以下面这部动漫视频为例，该视频长达20分钟，大小“高达”100MB，此时虽然表面上传输成功，但不同程度地出现了错误。bbb主机虽然可以播放画面，但是播放期间声音会突然消失或者发出噪音，aaa主机则只能播放前几分钟的画面，之后会突然卡住，屏幕变白（图8）。

究其原因，可以发现的是，两个接收到的文件均比原文件小（原大小118MB，下载所得的一个116MB，一个114MB），所以显然是发生了丢包。本次实验中，每次发送一个报文后调用usleep()方法使线程睡眠25毫秒，但这个延迟时间还是太短了，由于发送速率过快，发送方的缓冲区还没来得及发送出去就被覆写。UDP是无状态连接，没有专门的管线进行传输控制，无法检测丢包。

为了保证传输完整性，可做出如下调整：一是增大报文间发送的间隔时间，但是这个方法治标不治本，可以预见的是，文件大小越大，所需的间隔时间可能就会越长，当然极限是IO的读取速率，不过届时这个时间间隔已经过大以至于每次下

载文件所需的时间过长。

另一种方法是将文件切片，将每个文件切成较小的部分，对每个部分分别进行传输，最后在接收端再合成一个文件。事实上现在一些著名的视频直播网站如twitch、YouTube等就是采取的这种方法，直播时每个文件只包含几个帧，这样即使是个别分片出现了丢包情况也没有关系，直接将后面的一个分片接上或适当添加一些马赛克（经常看直播的朋友应该会体会到），整体观看效果还是可以保证的。

总之，UDP作为一种不可靠连接，用于传输文本效率还是非常可观的，而且通过本实验也发现，多播的灵活性非常高。当传输的报文本身不大时，多播模式的威力还是很大的，当所有人处于同一局域网，十分推荐采用多播模式进行实时通讯。但是，当可靠性不可忽略的场合，如文件传输，此时UDP的丢包将不可忽略，虽然可采取切片的方式进行改善，但是仍不推荐用UDP传输必须确保完整性的文件（如rar、apk等）。保证传输可靠性或传输大文件，还是烦请您采用面向连接的TCP协议。

参考文献

[1] Silke Feldmann, Kyandoghere Kyamakya, Ana Zapater, and Zighuo Lue. An indoor bluetooth-based positioning system: Concept, implementation and experimental evaluation. In *International Conference on Wireless Networks*, 2003.

[2] Carlos E. Galván-Tejada, José C. Carrasco-Jiménez, and Ramon F. Brena. Bluetooth-wifi based combined positioning algorithm, implementation and experimental evaluation. *Procedia Technology*, 7:37 – 45, 2013. 3rd Iberoamerican Conference on Electronics Engineering and Computer Science, CIIECC 2013.