Reliable Data Transfer

CS305计算机网络 Project答辩汇报

- 11811721庄湛
- 11911839聂雨荷

项目概要

- 计算机网络中网络层，链路层，物理层相对简单在不可靠信道上传输，导致数据传输是不可靠的。为了保证数据传输的可靠性，选择在运输层采用复杂的可靠数据传输协议(Reliable Data Transfer)，以确保网络的可靠性。
- 经典的RDT传输包括GBK(Go Back N)和选择重传两种方法，而互联网常用可靠传输协议TCP则是基于综合两者的优点实现的可靠信道传输。
- 我们小组通过学习TCP的传输机制及其特色，在大部分机制上实现与TCP相同的效果，完成了本次Project的设计。

设计思想

- 1.1 Message Format
- Organized our own packet
- 实现打包部分

SYN	FIN	ACK	seq	ack	ESE	CHEAKSUM	PAYLOAD
1bit	1bit	1bit	4bytes	4bytes	1bit	8bytes	LEN bytes

- 由于发送端将发送数据分片传送，我们额外增添了ECE bit位，用于标识每一次分片的结束

| 设计思想

- 1.2 Reliable Data Transfer
- 1. Accept and establish a connection
- 2. Maintain the connection, keep listening and replying
- 3. Close the connection, release the resource
- 实现经典的TCP状态机变换

客户端和服务器的3次握手连接

客户端和服务器连接后使用新的Socket和三个线程互相传输数据
— Send线程：只负责发送报文
— Recv线程：只负责接收报文
— Process线程：负责检测状态改变

客户端和服务器的4次挥手断开连接

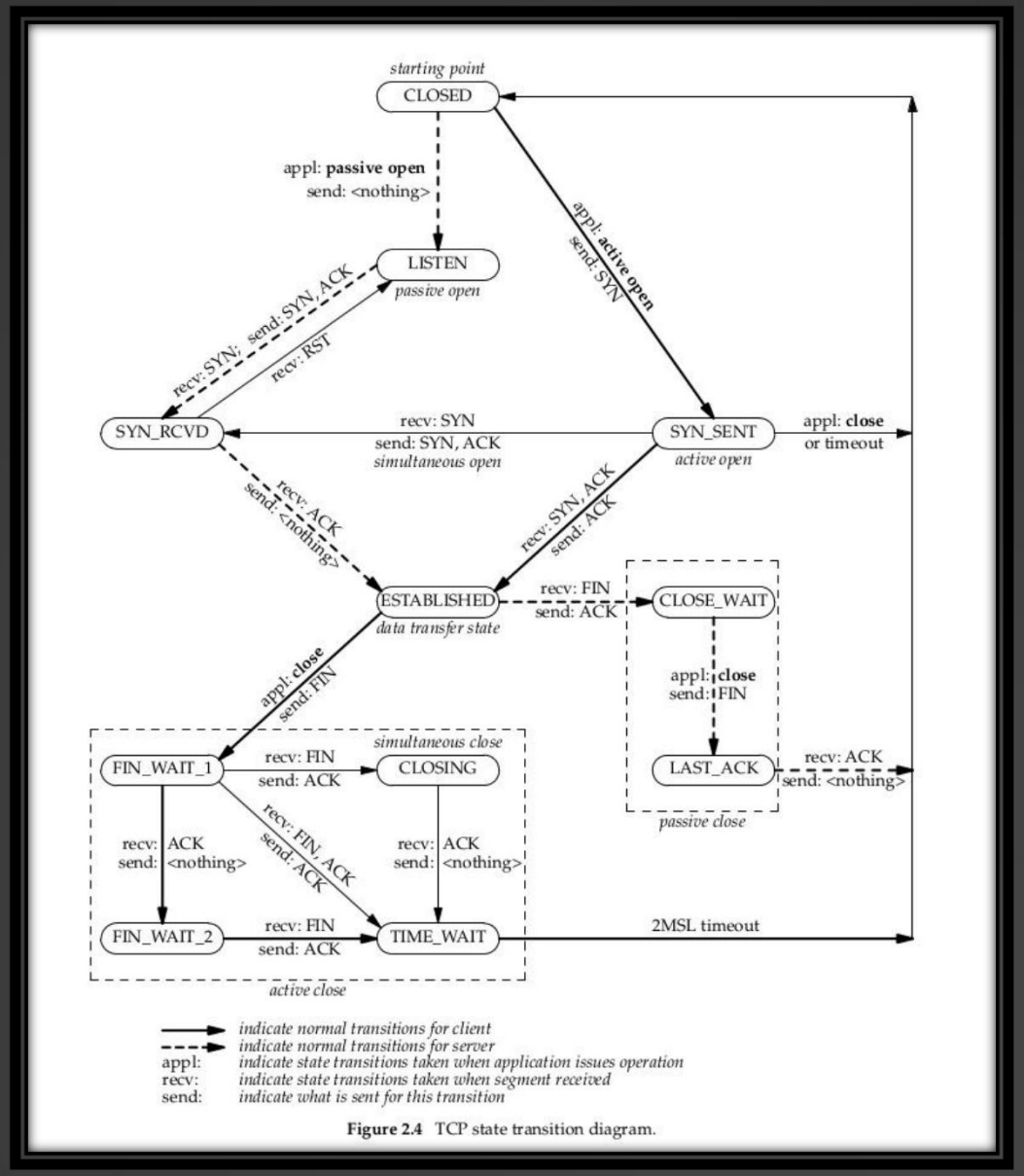

Figure 2.4 TCP state transition diagram.

| 设计思想

- 1.2 Reliable Data Transfer
- Your are responsible for detecting and correcting these abnormalities (through retransmission) when they occur.
- 1.Delay
- 2.Loss
- 3.Corrupt

延迟	接收端recvfrom()设置超时时间，如果超时重新发送ACK报文
丢包	接收端如果收到报文时，会做出判断： — 如果收到的SEQ > 需要的ACK，存入Buffer中，回复需要的ACK — 如果收到的SEQ < 回复需要的ACK，回复需要的ACK — 如果收到的SEQ = 需要的ACK，查看Buffer中的所有报文，将该报文与Buffer中可能存入的顺序满足的报文一并接收并且回复更新ACK
损坏	接收端通过CHECKSUM校验位进行校验，如果出现差错，则重新发送请求ACK报文

| 设计思想

- 1.3 Congestion Control
- You should implement congestion control in addition to the reliable transfer function to address the problem described and achieve a reasonable effective throughput as a fraction of R. 1.Delay
- 拥塞控制
- 实现思路参考TCP拥塞控制方法
- Send线程维护拥塞窗口，并且动态更新

功能实现

- Packet.py——负责对报文各种处理

核心函数	功能
__str__	重写方法，展示报文头的各个字段信息
get_xxx/set_xxx	获取某一字段的信息/设置某一字段的信息
compute_checksum	计算校验位
check	检验报文是否出错
set_bytes_from_string	将报文转化为可以设置字段信息的模式
get_bytes	将报文转化为发送的byte形式

功能实现

- FSM.py——状态机的转化以及连接建立和结束时的报文处理

11种状态	
CLOSED_Transition	LAST_ACK_Transition
LISTEN_Transition	FIN_WAIT_1_Transition
SYN_SENT_Transition	FIN_WAIT_2_Transition
SYN_RCVD_Transition	CLOSING_Transition
ESTABLISH_Transition	TIME_WAIT_Transition
CLOSE_WAIT_Transition	

| 功能实现

- rdt.py——实现可靠信道传输

核心函数	功能
connect	发起RDT连接
bind	绑定Socket地址
accept	处理connect连接并且返回新的连接Socket
close	关闭RDT连接
recv_from	recv	展示接受到的报文	接收数据
send_to	send	resend	展示发送的报文	发送数据	重传数据
thread_start	ESTABLISH阶段建立三个线程，具有不同功能
send/recv/proc_thread_method	三个不同线程的处理方法
move_window	调整拥塞控制的窗口变化
send_seq_wanted_packet	处理丢包，发送ACK

困难及解决方案

- 1.客户端和服务器互为收发对象，如何控制两边正常通讯？
- 当Socket处于ESTABLISH阶段时，使用3(+n)个线程分别做不同的处理
- 2.如何对于延时、丢包、损坏的进行处理？

延迟	send线程发送一个报文后，会打开time线程并将该报文传入time线程中，time线程自动计时，当超过指定时间后会重新发送该报文，当recv线程接收该报文的对应的ACK报文，会把time线程关闭
丢包	recv线程如果收到报文后，proc会做出判断，之后发送ACK报文： — 如果收到的SEQ > 需要的ACK，存入Buffer中，重新发送ACK — 如果收到的SEQ < 需要的ACK，重新发送ACK — 如果收到的SEQ = 需要的ACK，查看Buffer中的所有报文，将该报文与Buffer中可能存入的顺序满足的报文一并接收，更新ACK并发送
损坏	recv线程如果收到报文后，proc会检查报文是否有损毁，如果有，proc线程重新发送ACK

| 困难及解决方案

- 3.如何处理同时间服务器与多个客户端通讯？
- 设置同一时间内服务器只能与一个客户端连接，当三次握手结束后服务器新建一个新的socket连接，并返回到LISTEN状态。新的socket连接进入ESTABLISH状态，执行收发任务。
- 4.Seq_num, ACK_num如何动态更新，并且保证正确？
- 花费了大量时间进行微调和debug，保证了一次RDT中信息传输的准确性

| 测试效果

网络状态 (2份文件，297078bit)	时间 (s)
理想状态	12.1989805s
rate = 10000	100.92586139999999s
loss_rate = 0.00001， corrupt_rate = 0.00001	27.542554799999998s
loss_rate = 0.00005， corrupt_rate = 0.00005	60.806175200000006s
rate = 10000，loss_rate = 0.00001，corrupt_rate = 0.00001	128.05947579999997s