传输层的可靠数据传输协议设计示例

基于不可靠的网络数据通道的传输协议

新增三种功能

- 差错检测手段（软件校验和等）
- 接收端的反馈，告知接收情况
- 发送端重发机制，重传输出错数据

可靠数据传输协议rdt2.0

假设

① 报文在传输过程中可能出错
② 报文在传输过程中不会丢失
③ ACK和NAK在传输过程中不会出错
④ ACK和NAK在传输过程中不会丢失

- 携带传输层报文的包可能在传输过程中被损坏，导致收到的传输层报文出现错误
- 传输层报文不会在传输过程中被丢失
- 接收端通过反馈机制向发送端报告接收状况

函数名	功能说明
rdt_send(data)	发送数据data
rdt_rcv(data)	接收数据data
isACK(rcvpkt)	接收报文是否为肯定确认
isNAK(rcvpkt)	接收报文是否为否定确认
corrupt(rcvpkt)	接收报文在传输中是否出错
notcorrupt(rcvpkt)	接收报文在传输中是否无错

注意：示例中略去了端口号分配等事宜

可靠数据传输协议rdt2.0——发送端

发送端

初始化

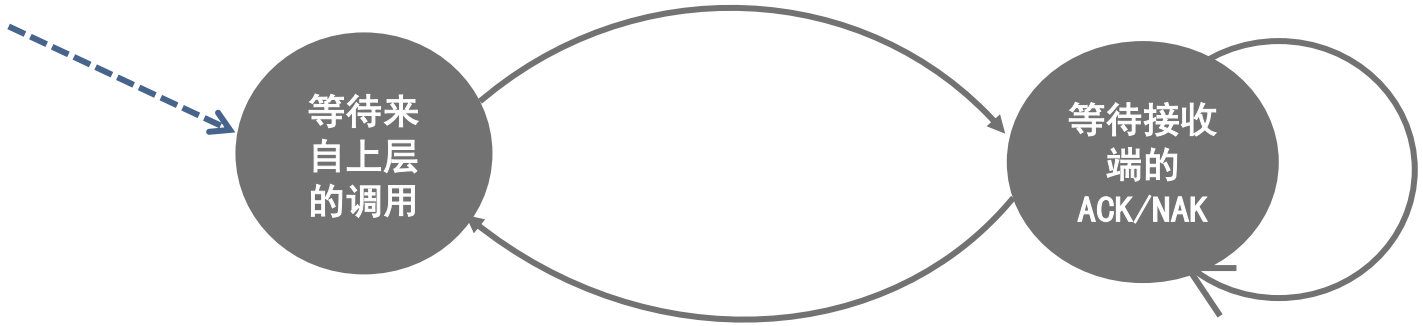
rdt_send(data) ①

sndpkt=make_pkt(data,checksum)

udt_send(sndpkt)

① 发送端将应用进程数据封装成报文发送：

- 为协助接收端校验是否出现错误报文中要加入校验和
- 调用发送函数发送报文
- 等待接收端的反馈



rdt_rcv(rcvpkt)&&isNAK(rcvpkt)

udt_send(sndpkt) ②

rdt_rcv(rcvpkt)&&isACK(rcvpkt)

Λ ③

③如果收到接收端反馈的肯定确认，说明包已被接收端完美接收：

- 返回初始状态，等待上层应用进程的调用

② 如收到接收端反馈的否定确认，说明包在传输中被破坏需重发：

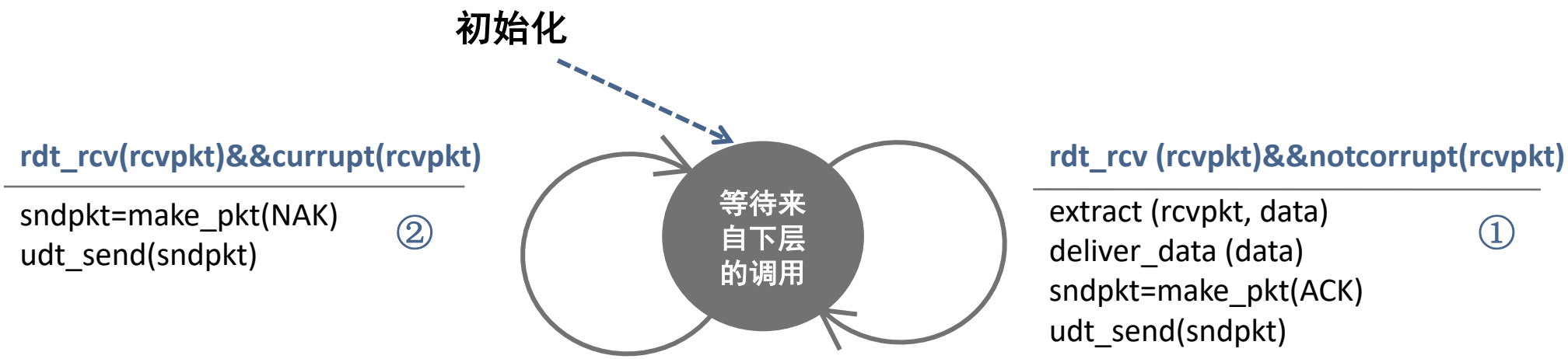
- 调用数据发送函数重发包
- 等待接收单的反馈

事件 / 动作

定义了当“事件”发生后，传输层实体采取的“动作”

可靠数据传输协议rdt2.0

接收端



②收到的报文被校验出有错，立即给发送端反馈否定确认，等待报文的重传。

①从校验正确报文中取出包含的数据交给上层应用进程，并给发送端反馈肯定确认后进入等待接收状态。

可靠数据传输协议rdt2.1

假设

① 报文在传输过程中可能出错
② 报文在传输过程中不会丢失
③ ACK和NAK传输可能出错
④ ACK和NAK不会被丢失

- 接收端要检查数据报文的正确性
- 发送端要检查确认信息是否正确。

函数名	功能说明
rdt_send(data)	发送数据data
rdt_rcv(data)	接收数据data
isACK(rcvpkt)	接收报文是否为肯定确认
isNAK(rcvpkt)	接收报文是否为否定确认
corrupt(rcvpkt)	接收报文在传输中是否出错
notcorrupt(rcvpkt)	接收报文在传输中是否无错

注意：为简单起见这里给出的ACK号为接收到的包序号

可靠数据传输协议rdt2.1——发送端

发送端

初始化

等待来自上层的调用0

rdt_send(data)

sndpkt=make_pkt (0,data,checksum)

udt_send(sndpkt)

①

①将上层数据封装成P0发送

等待来自接收端ACK或NAK0

rdt_rcv(rcvpkt)&&(corrupt(rcvpkt)||isNAK(rcvpkt))

udt_send(sndpkt)

②

②收到出错报文或否定确认，重发P0

③收到肯定确认则等待上层调用

rdt_rcv(rcvpkt)&&notcorrupt(rcvpkt&&isACK(rcvpkt))

Λ

③

等待来自上层调用1

④将上层数据封装成P1发送

rdt_send(data)

sndpkt=make_pkt (1,data,checksum)

udt_send(sndpkt)

④

等待来自接收端ACK或NAK1

rdt_rcv(rcvpkt)&&(corrupt(rcvpkt)||isNAK(rcvpkt))

udt_send(sndpkt)

⑤

⑤收到出错报文或者否定确认，重发P1

⑥收到ACK则等待上层

rdt_rcv(rcvpkt)&&notcorrupt(rcvpkt&&isACK(rcvpkt))

Λ

⑥

两次发送报文的处理过程完全一样，上面两状态和下面两状态的唯一区别在于发送报文的序号不同。

可靠数据传输协议rdt2.1——接收端

② rdt_rcv(rcvpkt)&&corrupt(rcvpkt)

sndpkt=make_pkt(NAK,checksum)
udt_send(sndpkt)

②校验有错反馈NAK

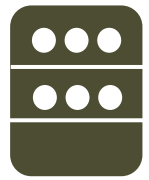
① rdt_rcv(rcvpkt)&&notcorrupt(rcvpkt)&&has_seq0(rcvpkt))

extract(rcvpkt,data)
deliver_data(data)
sndpkt=make_pk(ACK, checksum)
udt_send(sndpkt)

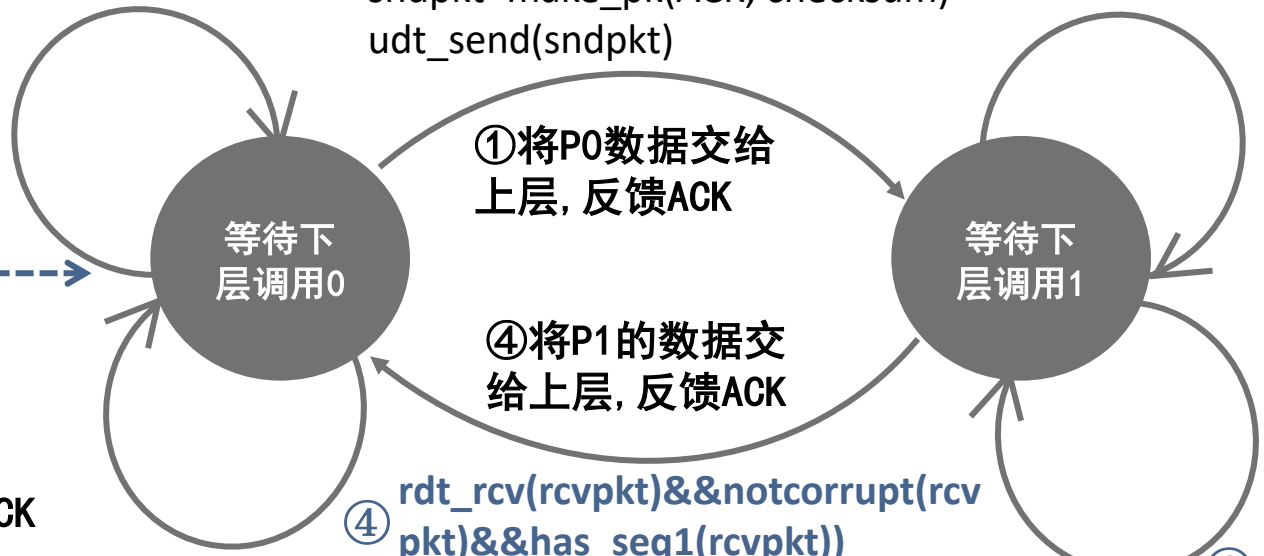
⑤ rdt_rcv(rcvpkt)&&(corrupt(rcvpkt)

sndpkt=make_pkt(NAK,checksum)
udt_send(sndpkt)

⑤校验有错则反馈NAK



接收端



在两种状态下针对收到的报文处理规则一摸一样，表现为左边1、2、3和右边的4、5、6完全一样。

③对重复包则重发ACK

③ rdt_rcv(rcvpkt)&&notcorrupt(rcvpkt)&&has_seq1(rcvpkt))

sndpkt=make_pkt(ACK,checksum)
udt_send(sndpkt)

④ rdt_rcv(rcvpkt)&&notcorrupt(rcvpkt)&&has_seq1(rcvpkt))

extract(rcvpkt,data)
deliver_data(data)
sndpkt=make_pk(ACK, checksum)
udt_send(sndpkt)

⑥对重复包则重发ACK

⑥ rdt_rcv(rcvpkt)&&notcorrupt(rcvpkt)&&has_seq0(rcvpkt))

sndpkt=make_pkt(ACK,checksum)
udt_send(sndpkt)

可靠数据传输协议rdt2.2

假设

① 报文在传输过程中可能出错和丢失

② 肯定确认ACK传输可能出错和丢失

- 用一种只有肯定确认机制完成可靠传输
- 接收端必须给出ACK号
- 发送端必须检查收到的ACK号

函数名	功能说明
rdt_send(data)	发送数据data
rdt_rcv(data)	接收数据data
isACK(rcvpkt)	接收报文是否为肯定确认
corrupt(rcvpkt)	接收报文在传输中是否出错
notcorrupt(rcvpkt)	接收报文在传输中是否无错

注意：为简单起见这里给出的ACK号为接收到的包序号

可靠数据传输协议rdt2.2——发送端

发送端

初始化

等待来自上层调用0

rdt_send(data) ①

sndpkt=make_pkt (0,data,checksum)

udt_send(sndpkt)

①将上层数据封装成P0发送

等待来自接收端ACK 0

② rdt_rcv(rcvpkt)&&(corrupt (rcvpkt)||isACK(rcvpkt,1))

udt_send(sndpkt)

②收到出错报文或确认P1，重发P0

③ rdt_rcv(rcvpkt)&&notcorrupt (rcvpkt&&isACK(rcvpkt， 0)

Λ

③收到确认P0，无需动作，等上层调用

等待来自上层调用1

④将上层数据封装成P1发送

rdt_send(data) ④

sndpkt=make_pkt (1,data,checksum)

udt_send(sndpkt)

等待来自接收端ACK1

⑤ rdt_rcv(rcvpkt)&&(corrupt (rcvpkt)||isACK(rcvpkt,0))

udt_send(sndpkt)

⑤收到出错报文或确认P0，重发P1

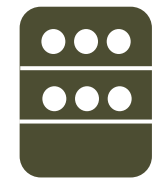
⑥ rdt_rcv(rcvpkt)&&notcorrupt(rcvpkt&&isACK(rcvpkt,1)

Λ

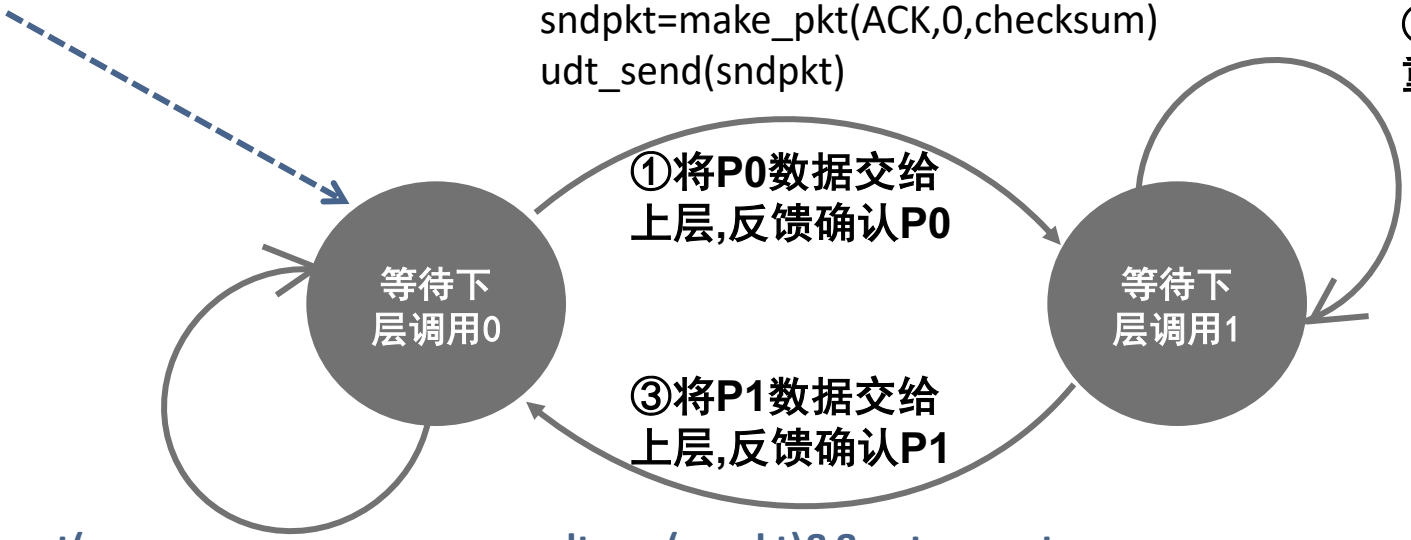
⑥收到确认P1，无需动作，等上层调用

上面两个状态和下面两个状态的唯一区别在于发送报文的序号是0还是1.

可靠数据传输协议rdt2.2——接收端



接收端



① rdt_rcv(rcvpkt)&&notcorrupt(rcvpkt)&&has_seq0(rcvpkt))

extract(rcvpkt,data)
deliver_data(data)
sndpkt=make_pkt(ACK,0,checksum)
udt_send(sndpkt)

② rdt_rcv(rcvpkt)&&(corrupt(rcvpkt)||has_seq0(rcvpkt))

udt_send(sndpkt)

②对出错报文或乱序报文，则重发确认P0

③ rdt_rcv(rcvpkt)&&notcorrupt(rcvpkt)&&has_seq1(rcvpkt))

extract(rcvpkt,data)
deliver_data(data)
sndpkt=make_pkt(ACK,1, checksum)
udt_send(sndpkt)

④ rdt_rcv(rcvpkt)&&(corrupt(rcvpkt)||has_seq1(rcvpkt))

udt_send(andpkt)

④对出错/乱序报文重发确认P1

在两种状态下针对收到的报文处理规则一摸一样的，表现为右上1、2和左下的3、4完全一样，只是序号从0换成了1。