

主讲人: 聂兰顺

本讲主题

Rdt 2.1和2.2



Rdt 2.0有什么缺陷?

- ❖ 如果ACK/NAK消息发生错误/被破坏(corrupted)会怎么样?
 - ▶ 为ACK/NAK增加校验和,检错并纠错
 - ➤ 发送方收到被破坏ACK/NAK时不知道接收方发生了什么,添加额外的控制消息
 - ➤ 如果ACK/NAK坏掉,发送方重传
 - ▶ 不能简单的重传: 产生重复分组
- ❖ 如何解决重复分组问题?
 - 序列号(Sequence number): 发送方给每个分组增加序列号
 - 接收方丢弃重复分组

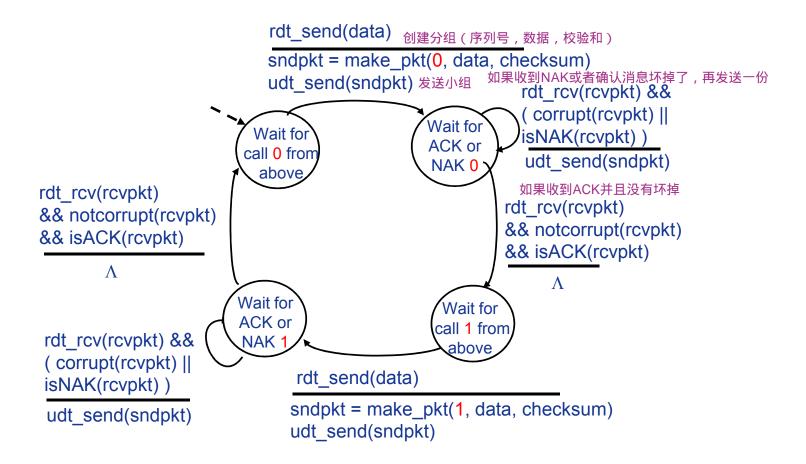
Sender sends one packet, then waits for receiver response





Rdt 2.1: 发送方, 应对ACK/NAK破坏







Rdt 2.1:接收方,应对ACK/NAK破坏

rdt rcv(rcvpkt) && notcorrupt(rcvpkt) && has seq0(rcvpkt) extract(rcvpkt,data) deliver data(data) sndpkt = make pkt(ACK, chksum) udt send(sndpkt) rdt rcv(rcvpkt) && (corrupt(rcvpkt) rdt rcv(rcvpkt) && (corrupt(rcvpkt) sndpkt = make pkt(NAK, chksum) udt send(sndpkt) Wait for Wait fo 0 from rdt rcv(rcvpkt) && 1 from rdt rcv(rcvpkt) && below not corrupt(rcvpkt) && below not corrupt(rcvpkt) && has seq1(rcvpkt) has seq0(rcvpkt) sndpkt = make pkt(ACK, chksum) sndpkt = make pkt(ACK, chksum) udt send(sndpkt) udt send(sndpkt) 送方重发分组0,而不是分组 rdt rcv(rcvpkt) && notcorrupt(rcvpkt) && has seq1(rcvpkt) extract(rcvpkt,data) deliver data(data) sndpkt = make pkt(ACK, chksum) udt send(sndpkt)



假如发送方无误的将分组0

发送给了接收方,接收方也

发送ACK分组准备接收分组

1,但是ACK分组却在传输

过程中损坏了,这会导致发

Rdt 2.1 vs. Rdt 2.0

❖发送方:

- □为每个分组增加了序列号
- □两个序列号(0, 1)就够用,为什么?
- □需校验ACK/NAK消息是否发生错 误
- □状态数量翻倍
 - □状态必须"记住""当前"的分组 序列号

*接收方

- □需判断分组是否是重复
 - □当前所处状态提供了期望收到分组 的序列号
- □注意:接收方无法知道ACK/NAK 是否被发送方正确收到





Rdt 2.2: 无**NAK**消息协议



- ❖ 我们真的需要两种确认消息(ACK + NAK)吗?
- ❖ 与rdt 2.1功能相同,但是只使用ACK
- ❖ 如何实现?
 - ▶ 接收方通过ACK告知最后一个被正确接收的分组
 - ➤ 在ACK消息中显式地加入被确认分组的序列号
- ❖ 发送方收到重复ACK之后,采取与收到NAK消息相同的动作
 - ▶ 重传当前分组



Rdt 2.2 FSM片段

