Reliable Data Transfer

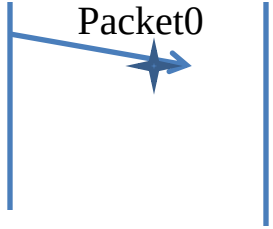
Kameswari Chebrolu

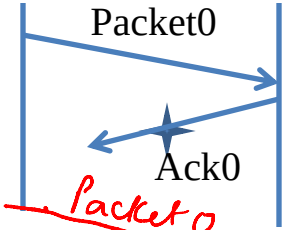
SNR

RDT: Channel with Errors and Losses

- Will RDTv2.1 work?
- Sender gets no feedback: Need a Timeout mechanism
- How long to wait?

↳ Link, Tx time, Prop time, Processing TO





Packet0

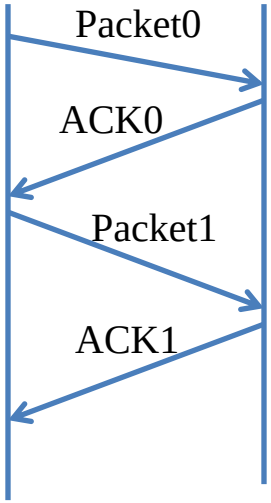
RDTv3.0

- Sender waits "Reasonable" amount of time for ACK
  - Retransmits if no ACK received in this time
- If pkt (or ACK) just delayed (not lost)
  - Retransmission will be duplicate, seq. #'s help resolve this

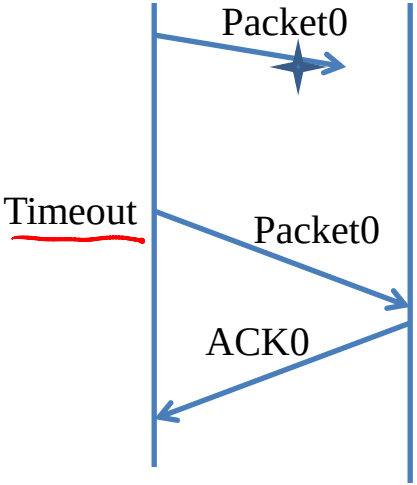
Required Functionality:

- Error Detection mechanism
  - Checksum, CRC etc
- Receiver Feedback
  - ACK + NACK
- Data Sequence Numbers
- ACK carries data seq. No.
- Timeout Timer

RDTv3.0: Stop and Wait Protocol In Action



(a) No Loss

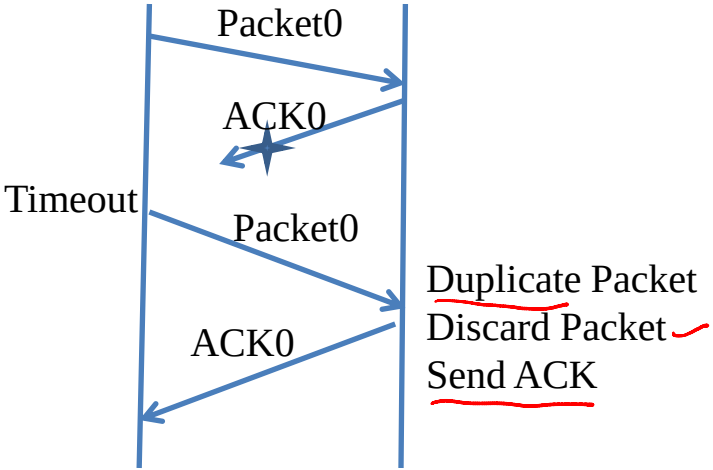


(b) Lost Packet

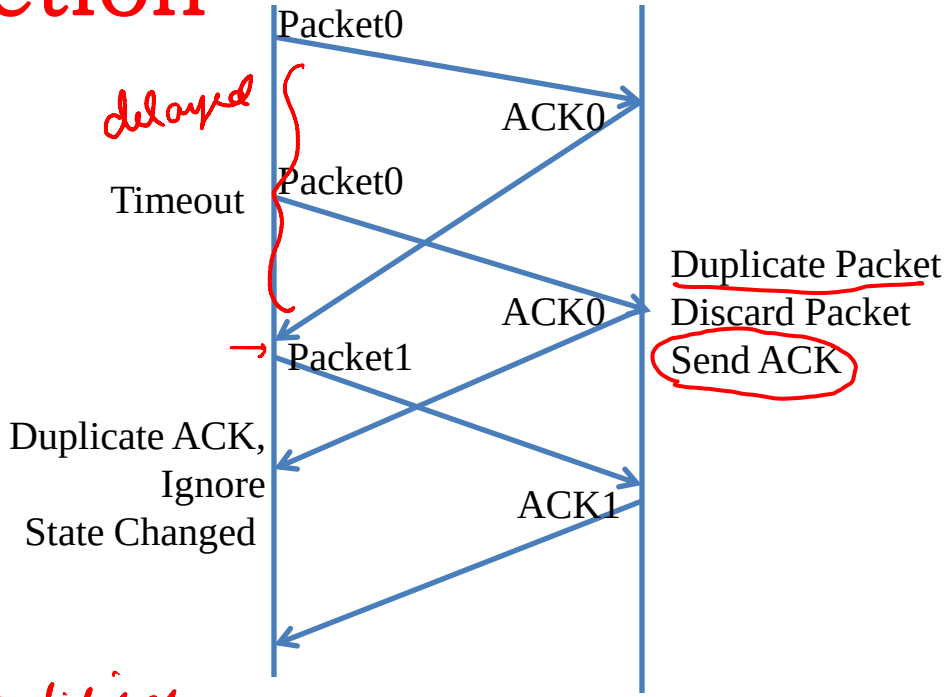
0, 1

Also called Alternating Bit Protocol

RDTv3.0: Stop and Wait Protocol In Action



(c) Lost ACK



RDTv2.1
↳ state didn't change

(d) Premature Timeout

Design of RDT protocols

- Many challenges to handle
  - ACKs/Packet loss, ACKs/Packet delayed, Duplication of packets, Reordering, Incorrect timeout timer settings, Receiver capabilities → buffer
  - Protocol has to work <u>correctly and efficiently</u> in spite of all this

TCP

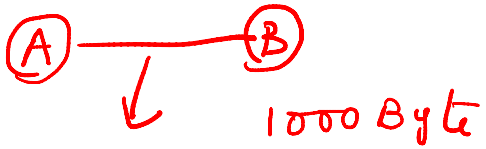
NACK vs ACK

x $x+1$ $x+2$ long time

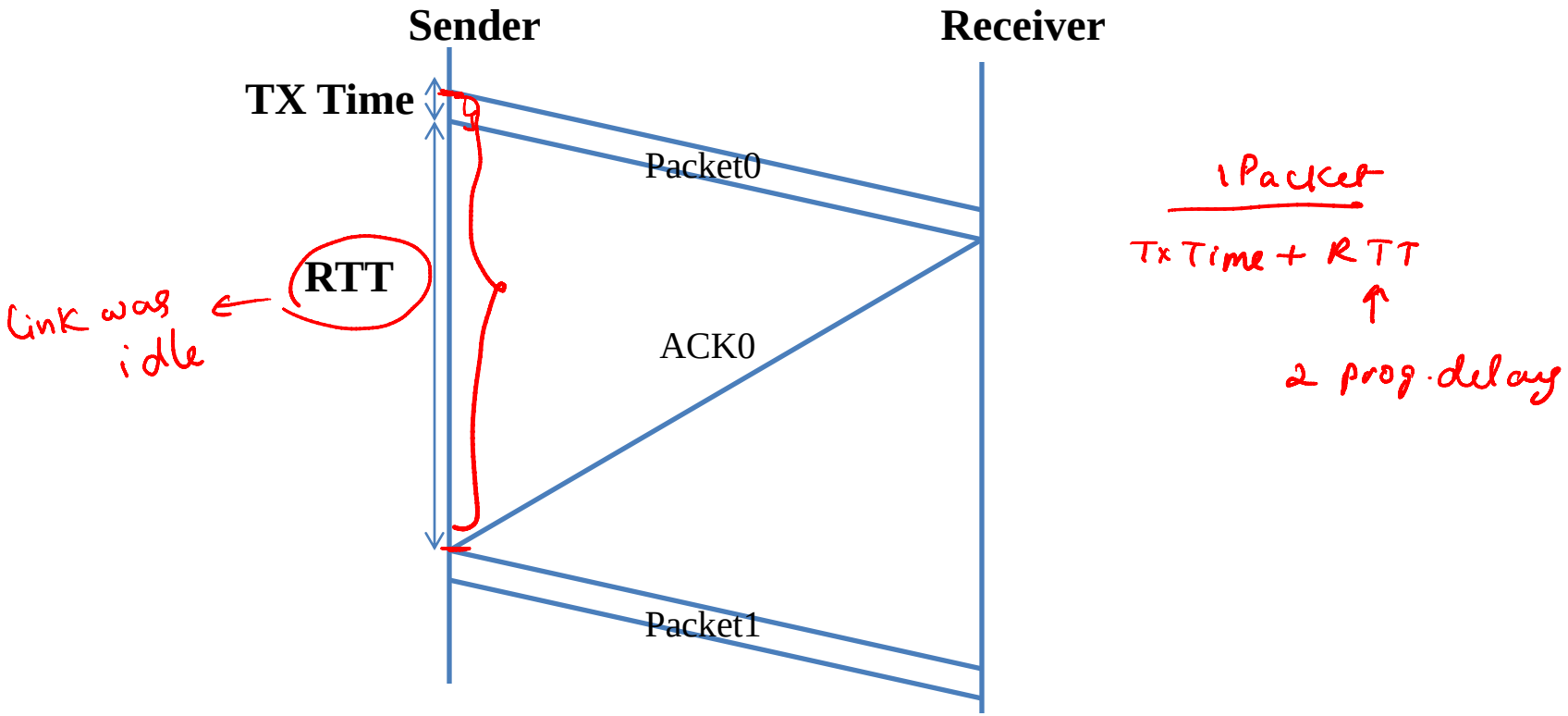
- Can conclude packet loss on detecting 'holes'
  - Long delay between some packets can slow down recovery
  - What if the last packet in the flow is lost? deadlock
    - Receiver doesn't generate NACK, sender assume 'all is well'
- Advantage of NACK: If errors are infrequent, reduces overhead of feedback NACK + ACK

RDT v3.0

Performance of Stop and Wait Protocol



- What is the achieved throughput?
  - 10 Mbps link, 10 ms prop. delay, 1KB packet, ACK too small (ignore its Transmission time)
- Throughput: 8000 bits / [(8000/10^7)+2*0.010] = 384.6Kbps
- Utilization = 384.6kbps/10000kbps = 3.8%



Utilization = Transmission time / (Transmission time + RTT)

Summary

- Reliable data transfer protocols provide 'reliable channel' service abstraction to higher layers
- We incrementally determined the required functionality needed in RDT protocols
  - bit error
  - losses
- The current protocol designed is inefficient
- Future: Build on this framework to design better protocols
  → functionality