COMP3331 Week 4 Lecture 2

Transport Layer Part 2

# 3.4 Principles of reliable data transfer

**rdt2.0: channel with bit errors**

* we use checksum to detect bit errors
* Recovering from error – feedback to sender with control msgs for retransmit(ACKS, NAKS)

Flaw – possibility of ACK/NAK corrupted, sender does not know what happened at receiver

Diagram

Description automatically generated

Convention we follow in this course

This is what we call the **stop and wait** protocol

**rdt 2.1: account for ACK/NAK corruption**

**sender:**

* add sequencer number to each packet
* two seq. #’s (0,1) will suffice (because we stop and wait)
* must check if received ACK/NAK corrupted

**Receiver:**

* must check if received packet is duplicate
* note: receiver cannot know if its last ACK/NAK received OK at sender

Diagram

Description automatically generated

**rdt2.2: a NAK-free protocol**

same functionality as rdt2.1, using ACKs only, the notion of **duplicate ack**

Diagram

Description automatically generated

**rdt3.0: channels with errors and loss - underlying channel can also lose packets**

how to achieve this? We use timer.

Diagram

Description automatically generated

Diagram

Description automatically generated

**Quiz:**

***Which of the following are needed for reliable data transfer with only packet corruption (and no loss or reordering)? Use only as much as is strictly needed.***

We need checksums to check error, we need acks to check whether a packet has been corrupted. We also need sequence number to check for duplicates.

Text, letter

Description automatically generated

***If packets (and ACKs and NACKs) could be lost which of the following is true of RDT 2.1 (or 2.2)?***

Text

Description automatically generated

***Which of the following are needed for reliable data transfer to handle packet corruption and loss? Use only as much as is strictly needed.***

Text, letter

Description automatically generated

This is rdt 3.0, we need timer in the event of packet loss.

# Rdt3.0

## stop and wait operation

Stop and wait operation – poor performance, limits use of resources

utilization of sender (Us)- fraction of time sender busy sending

A picture containing text, clock

Description automatically generated

## Pipelined protocols

pipelining – sender allows multiple, ‘in-flight’, yet-to-be-acknowledged pkts

Chart

Description automatically generated

# Go back N – sliding window protocol

* Sender can have up to N unacked packets in pipeline
* Sender has a **single timer** for oldest unacked packet, when timer expires, retransmit all unacked packets
* No buffer available at receiver – out of order packets are discarded
* Receiver only sends **cumulative ack**, doesn’t ack new pkt if there’s gap/loss
* When the oldest packet is received, the sliding window can move the window forward
* The sender has to resend all packets in the window if a packet is lost.

Text

Description automatically generated

# Selective Repeat

* Sender can have up to N unacked pkts in pipeline
* Sender **maintains timer for each unacked pkt**, when timer expires, retransmit only that unacked pkt
* Receiver has buffer – can accept out of order pkts
* Receiver sends **individual ack** for each pkt

Diagram

Description automatically generated

**Selective repeat window size relation**

Sender window size <= ½ of Sequence number space