

Week 5: TCP

CSCI 4431/6431 Computer Networks

Prof. Tim Wood

George Washington University

timwood@gwu.edu

<https://go.gwu.edu/csnetworks>

Today

- **Quiz** with your Project 1 team
- Finish **reliable data transfer**: rdt2.2, loss, timers, pipelining
- **TCP**: sequence numbers, timeouts, flow control, the handshake
- **Project 1**: meet your partner group, write Spec v2

Housekeeping

- **Spec v1** was due before class. Did you push the spec-v1/1 tag?
- **Partner groups** are listed on the **claim sheet**.
- Sit with your **team**.
- **Discord** and office hours for questions

Quiz

With your **Project 1** team

- It's the worksheet we didn't finish last week
- List names of all students present in class

Reliable Data Transfer

continued

Slides: https://gaia.cs.umass.edu/kurose_ross/ppt-9e/Chapter_3_v9.0.pptx

Chapter 3 - Slides 54 - 58

Last week ended on rdt2.0's fatal flaw: **a corrupted ACK leaves the sender blind.**

- It can't read the response, so it retransmits, risking a **duplicate**
- **rdt2.1**: a sequence number on every packet, 0 and 1
- **rdt2.2**: Replace NAK by including sequence number with ACK

Chapter 3 - Slides 59 - 64

- **rdt3.0**: the channel can now **lose** packets as well as corrupt them
- The sender waits a "reasonable" time, then retransmits if no ACK
- Premature timeout → **duplicate**, and sequence numbers already handle that

Chapter 3 - Slides 65 - 69

- Stop-and-wait on a 1 Gbps link, 30 ms RTT: $U = 0.00027$
- The protocol, not the link, is the bottleneck
- **Pipelining:** keep several packets in flight at once to improve utilization

Chapter 3 - Slides 70 - 73

- **Go-Back-N:** window of N, **cumulative ACKs**, one timer, resend everything after a loss
- **Selective Repeat:** ACK each packet individually, buffer out of order, resend only what was lost
- The Go-Back-N and Selective Repeat walkthroughs (slides 74 - 78) are in video **3.4b**

TCP uses cumulative ACKs. Keep that in mind for the next section.

Connection-Oriented Transport

TCP (~4:15pm?)

Chapter 3 - Slides 80 - 81

- Point-to-point, reliable, in-order **byte stream**
- Full duplex, connection-oriented, flow controlled
- Pipelined, with window size set by flow and congestion control
- Segment header: **20 bytes** minimum (UDP's is 8)

Every rdt idea, in one header

rdt mechanism	Where it lives in TCP
Checksum (rdt2.0)	checksum field
Sequence numbers (rdt2.1)	sequence number, which counts bytes
NAK-free ACKs (rdt2.2)	ack number + A flag, cumulative
Timer (rdt3.0)	not in the header, it's sender state
Pipelining window	receive window (and a sender-side cwnd)

Chapter 3 - Slides 82 - 90

- **Seq #**: byte-stream number of the *first byte* in the segment
- **ACK #**: the *next byte expected* from the other side
- Cumulative: ACK 43 means "I have everything through 42"
- Timeouts need to be picked carefully based on RTT and its variation

The number that trips people up: an ACK names what you want next, not what you just got.

Chapter 3 - Slide 91: fast retransmit

Don't wait for the timer.

Three duplicate ACKs → resend the oldest unACKed segment.

*Three **duplicate** ACKs means four ACKs for the same byte in total. Count carefully.*

Break (~4:36pm)

5 minute break

When you come back, sit near your partner group.

Claim sheet lists who your partner group is.

Work on TCP worksheet depending on timing

Flow Control and Connection Management

Chapter 3 - Slides 96 - 97

- Network delivers faster than the app reads → the socket buffer fills
- **Flow control:** the *receiver* controls the *sender*
- Receiver advertises free buffer space in **rwnd**
- Sender keeps in-flight data $\leq$ rwnd

Chapter 3 - Slides 99 - 103

- Before data: agree to talk, and agree on starting sequence numbers
- **Will a 2-way handshake work?** "Let's talk." "OK."
 - **Half-open connection:** a retransmitted request arrives after the client is gone
 - **Duplicate data accepted:** an old retransmission gets replayed into a new connection

Think about: variable delay, retransmitted requests, a side you can't see.

Chapter 3 - Slides 104 - 106

- **SYN** (seq = x) → **SYN-ACK** (seq = y, ack = x+1) → **ACK** (ack = y+1)
- A SYN carries no data but **consumes one sequence number**
- *On belay? Belay on. Climbing.*
- Closing: each side sends its own **FIN**, and each FIN is ACKed

Wrap up and Homework

- Spec v2 due Wed 9/30, 11:59 PM: tag spec-v2/1
- Watch 3.6 and 3.7 — congestion control, which we cover next week. Come having seen it
- Optional: 3.8 on QUIC (very interesting protocol!)
- Next week: guest lecture, congestion control, and midterm review
- Midterm: 10/7 - you can bring your quizzes/worksheets + 1 page of handwritten notes

Project 1

Merging Specifications

Phase 2: the rules

- This week is the **only** time you may discuss the spec with your partner group
- Both groups implement from **Spec v2 and nothing else**
- Same assumption as before: the network is **reliable**, with no loss, no reordering and no duplicates
- The **disagreement table** is a major deliverable, not a preamble

Template: `spec/spec-v2.md` in your repo

The next 60 minutes

Min	Do
0	One person opens a shared <code>spec-v2.md</code> or google doc.
1	Read the other group's v1.
15	Disagreement table. Message format first, then message types, then errors.
45	Write one full exchange, byte for byte , that both groups agree on.
55	Still unresolved. Write down what you couldn't settle.

Settle the format first

Two implementations that disagree on *meaning* still exchange messages.

Two that disagree on *bytes* can't exchange anything.

- Text or binary? Delimiters? Field order?
- How does a receiver know where a message ends?
- What exactly does an error reply look like?

Before you leave today

1. A **disagreement table** with a row for every difference you found
2. **One agreed exchange**, written out byte for byte
3. A "still unresolved" list

Finish Spec v2 by 9/30. After that date you can no longer communicate with your partner team.

Chapter 3 Resources

- Textbook Slides
- Practice Problems: c3q3, c3q4 and c3q6 are TCP; c3q5 is congestion control
- Videos (Full playlist):
 - 3.4 Reliable Data Transfer, part 1 and part 2
 - 3.5 TCP, part 1 and part 2
 - 3.6 Principles of Congestion Control
 - 3.7 TCP Congestion Control
 - 3.8 Evolution of the Transport Layer
 - 3.9 Summary