Transport Layer: TCP and UDP

Raj Jain
Washington University in Saint Louis
Saint Louis, MO 63130
Jain@wustl.edu

Audio/Video recordings of this lecture are available on-line at:

http://www.cse.wustl.edu/~jain/cse473-25/

Student Questions

Overview

- ❑ Transport Layer Design Issues:
  - Multiplexing/Demultiplexing
  - Reliable Data Transfer
  - Flow control
  - Congestion control
- ❑ UDP
- ❑ TCP
  - Header format, connection management, checksum
  - Congestion Control
- ❑ **Note**: This class lecture is based on Chapter 3 of the textbook (Kurose and Ross) and the figures provided by the authors.

Student Questions

- ❑ I am still a bit confused about the acknowledgment number/ACK in general. When is the ACK field seqnum +1, and when is it equal to seqnum?

It depends on the transport protocols. TCP designers decided to make "Ack n" mean "I have received n-1st byte, and I am waiting for nth byte." In most other protocols, "Ack n" means "I have received nth packet, and I am waiting for n+1st packet".

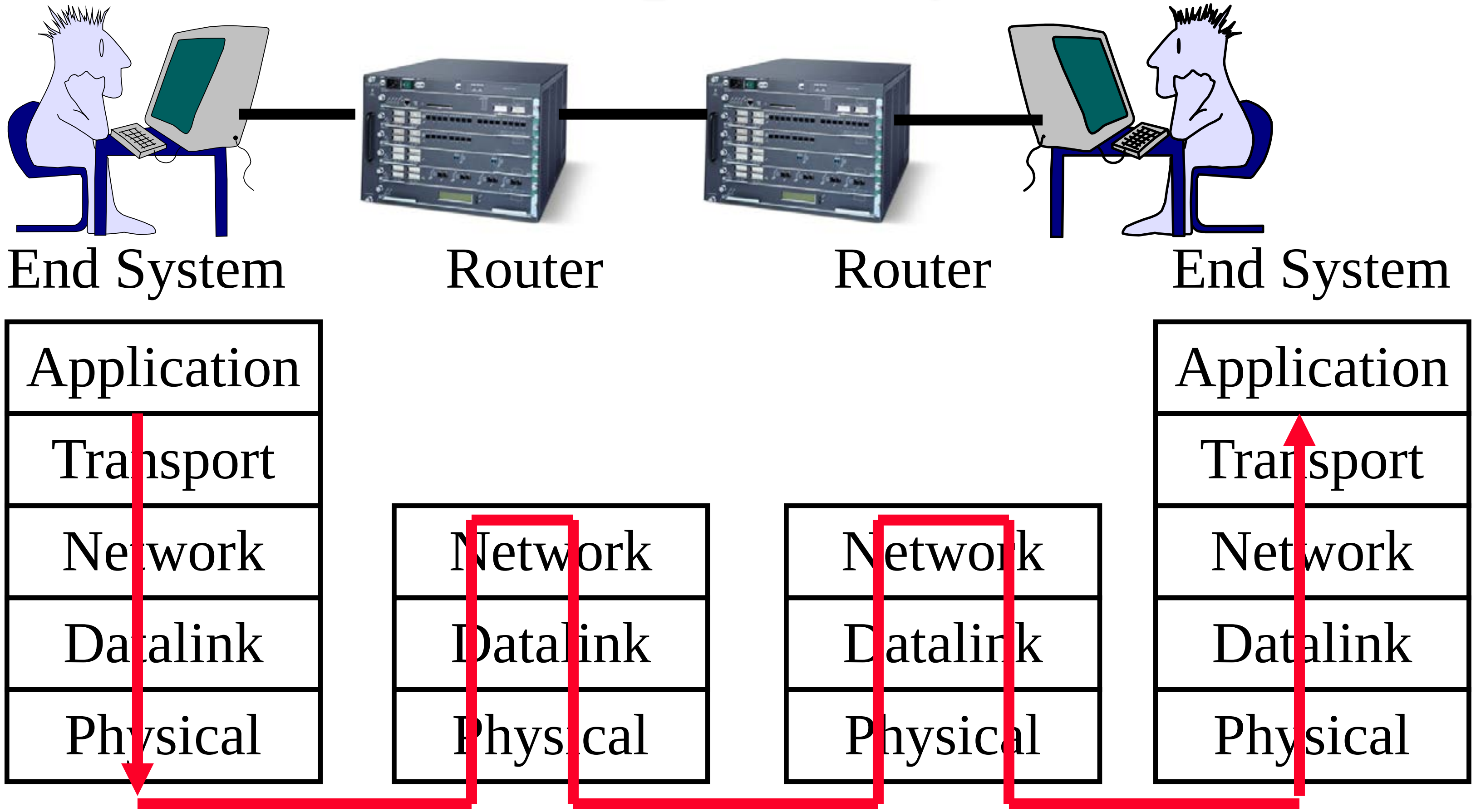
Transport Layer Design Issues

1. Transport Layer Functions
2. Multiplexing and Demultiplexing
3. Error Detection: Checksum
4. Flow Control
5. Efficiency Principle
6. Error Control: Retransmissions

Student Questions

- What is the difference between SDUs and PDUs?

Transport Layer



End System	Router	Router	End System
Application			Application
Transport			Transport
Network	Network	Network	Network
Datalink	Datalink	Datalink	Datalink
Physical	Physical	Physical	Physical

- Transport = End-to-End Services
  Services required at source and destination systems
  Not required on intermediate hops

Student Questions

- ❑ What do buffers refer to?

Incoming and outgoing packets are stored in the memory. That part of the memory is called a buffer.

- ❑ Does this mean that routers cannot do Flow control & Loss detection & Congestion Control since it doesn't have a Transport layer?

Technically yes. But it can react to its own buffer overflow.

- ❑ What do hops mean? *1 Hop = 1 Link*
- ❑ What does the program running on the router look like? Is it a Linux-based program?

Proprietary OS. It could be Linux. There is no standard.

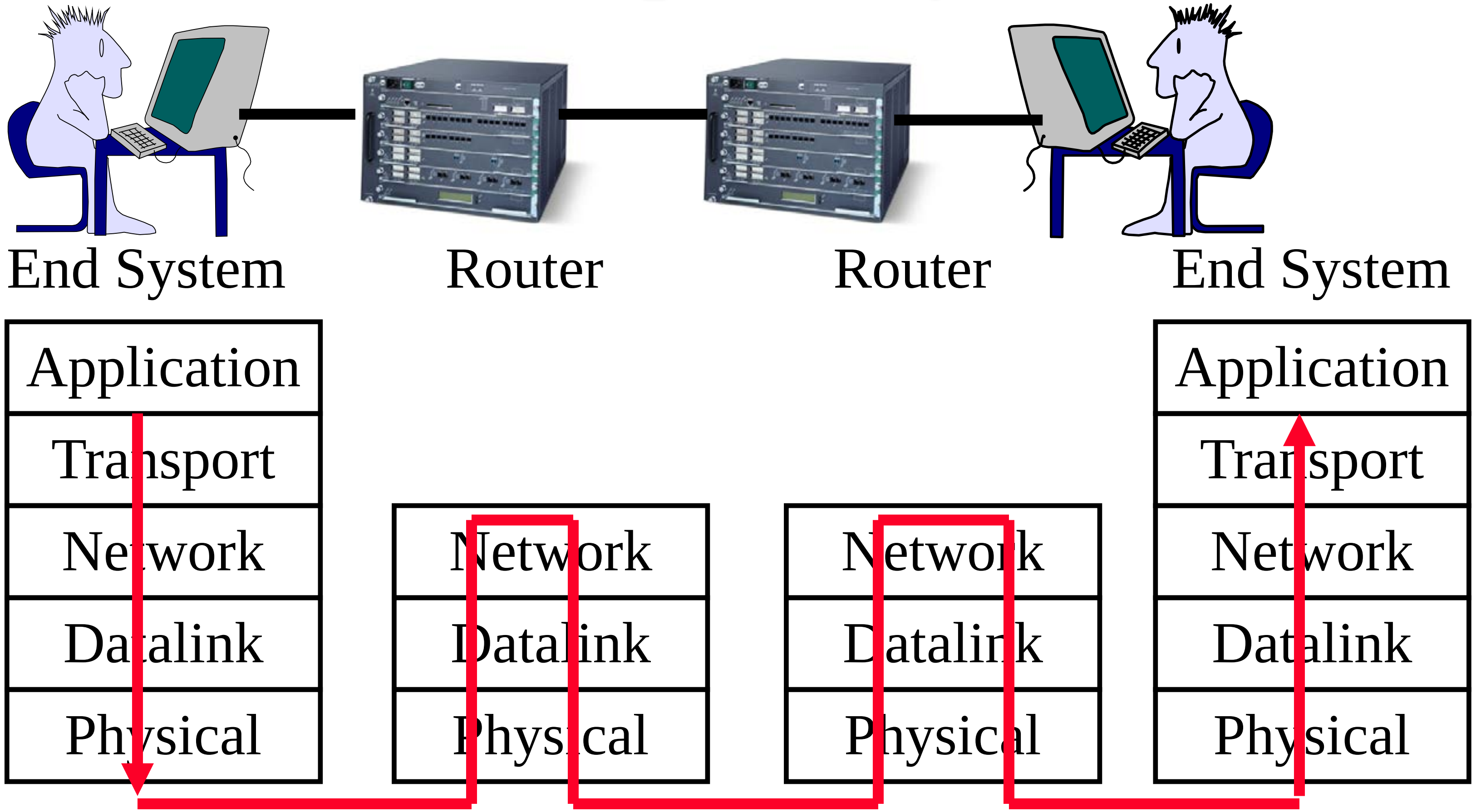
- ❑ What hardware does Network Interface Controller (NIC) run on?

NIC is the network card connected to the CPU. For example, USB dongles for Ethernet are NICs.

- ❑ Is port 80 here a socket port? Is a socket just an address plus a port?

Please see the Chapter 2 Q&A video and slides.

Transport Layer



End System

Router

Router

End System

End System	Router	Router	End System
Application			Application
Transport			Transport
Network	Network	Network	Network
Datalink	Datalink	Datalink	Datalink
Physical	Physical	Physical	Physical

- Transport = End-to-End Services
  Services required at source and destination systems
  Not required on intermediate hops

Student Questions

- When a packet goes through a router, does the router check the layer-4 header and layer-5 header of the packet? Since the layer-4 header and layer-5 header do not contain IP addresses, is the routing process done in layer-3 by a router?

Yes, the router looks only at the layer-3 header. Layer 4-5 headers are part of the data field and are not interpreted by the network layer. The network layer is responsible for getting the packet to the destination IP address. Therefore, routers are responsible for routing.

- Are headers of packets removed and added in each router? *Each layer updates only its header at each node. Routers do not change TCP or application headers.*
- Does this mean that if I invent my transport protocol, it will also work on the Internet?

Yes, except that TCP/IP is not strictly layered. TCP and IP are tightly coupled. Any change in one requires changing the other. So you will have to worry about being compatible with IP.

Transport Layer Functions

1. **Multiplexing and demultiplexing**: Among applications and processes at end systems
2. **Error detection**: Bit errors
3. **Loss detection**: Lost packets due to buffer overflow at intermediate systems (Sequence numbers and acks)
4. **Error/loss recovery**: Retransmissions
5. **Flow control**: Ensuring the destination has buffers
6. **Congestion Control**: Ensuring the network has capacity

Not all transports provide all functions

Student Questions

❑ What kinds of transports don't provide functions?

There are many types of transport. Some do not provide some functions. Each uses a different method for each function. This is a comprehensive but not complete list of functions provided by various transports.

❑ How does the computer know if bits or packets were lost if it doesn't have the original file to compare it to? Is it using the checksum?

A packet is received, but there is no buffer to store it.

❑ Can the intermediate nodes have any capability restrictions with regard to retransmitting data, or is it all the transport protocol?

No restrictions. Most wireless routers do retransmit. More about it is in Chapter 6 on Datalink.

❑ What transport does not provide one or more of the above functions?

UDP. See Slide 2-10.

Transport Layer Functions

1. **Multiplexing and demultiplexing**: Among applications and processes at end systems
2. **Error detection**: Bit errors
3. **Loss detection**: Lost packets due to buffer overflow at intermediate systems (Sequence numbers and acks)
4. **Error/loss recovery**: Retransmissions
5. **Flow control**: Ensuring the destination has buffers
6. **Congestion Control**: Ensuring the network has capacity

Not all transports provide all functions

Student Questions

□ Is this a comprehensive list of all possible functions?

No. This list is a good start.

□Are networks considered 'better" if they implement more functions?

No. More functions ⇒ More Delay ⇒ More Cost

□ Does congestion control include flow control?

No. But they are similar.

□ Is there a function that is mandatory for all transport?

No.

□ Does TCP check for duplicates (if a message is erroneously transmitted multiple times)?

Yes. Each byte is numbered.

□ What is a hop?

The link between two routers.

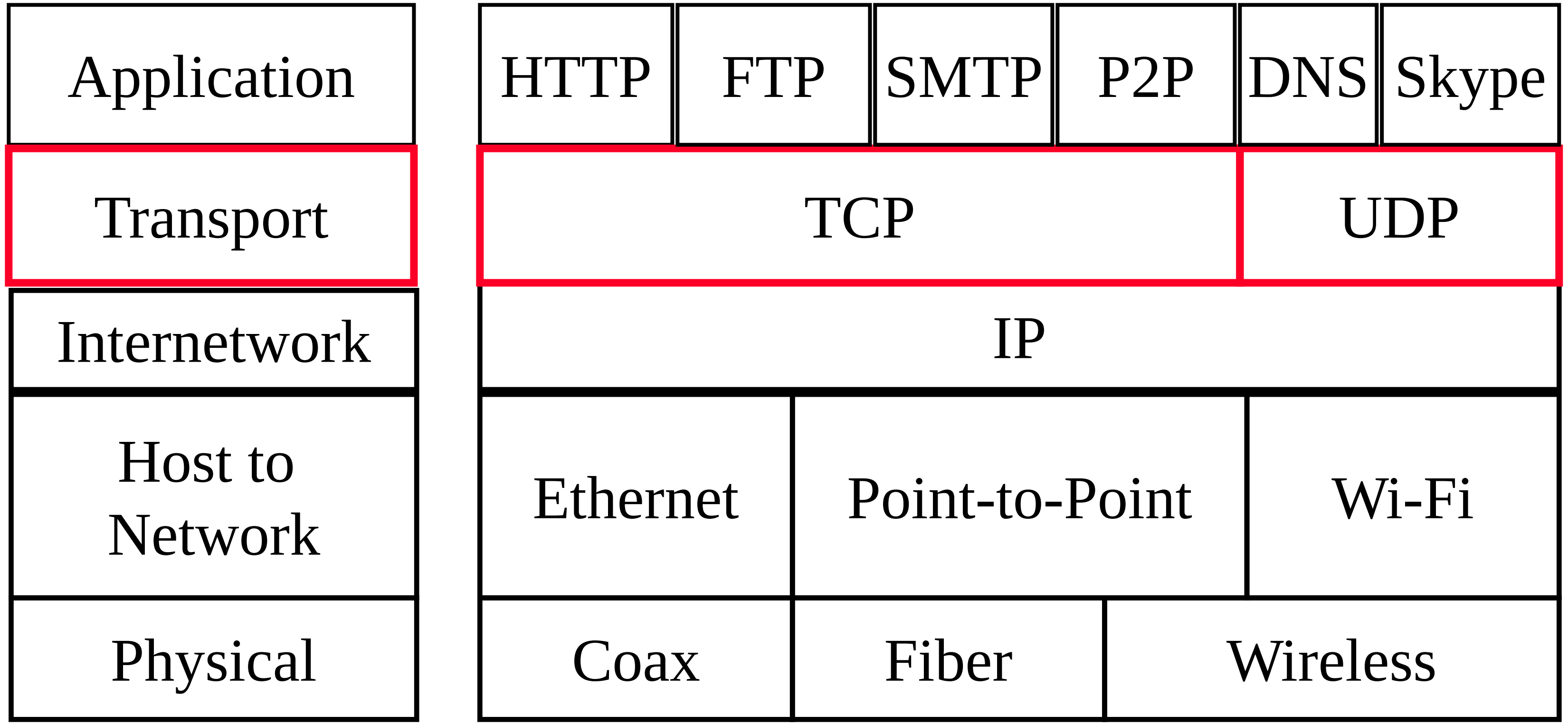
□ For congestion control, does the router want other routers to stop sending packets to it, or does it want the source to stop sending packets? *Source control works better.*

□ What would happen if some bit errors were not detected by the error detection?

The message will be incorrect.

Protocol Layers

- Top-Down approach



Application | HTTP | FTP | SMTP | P2P | DNS | Skype

Transport | TCP | UDP

Internetwork | IP

Host to Network | Ethernet | Point-to-Point | Wi-Fi

Physical | Coax | Fiber | Wireless

Student Questions

- Can an application send out a request using TCP (e.g., HTTP request) on one port number and request that the response is sent back to a different port number?

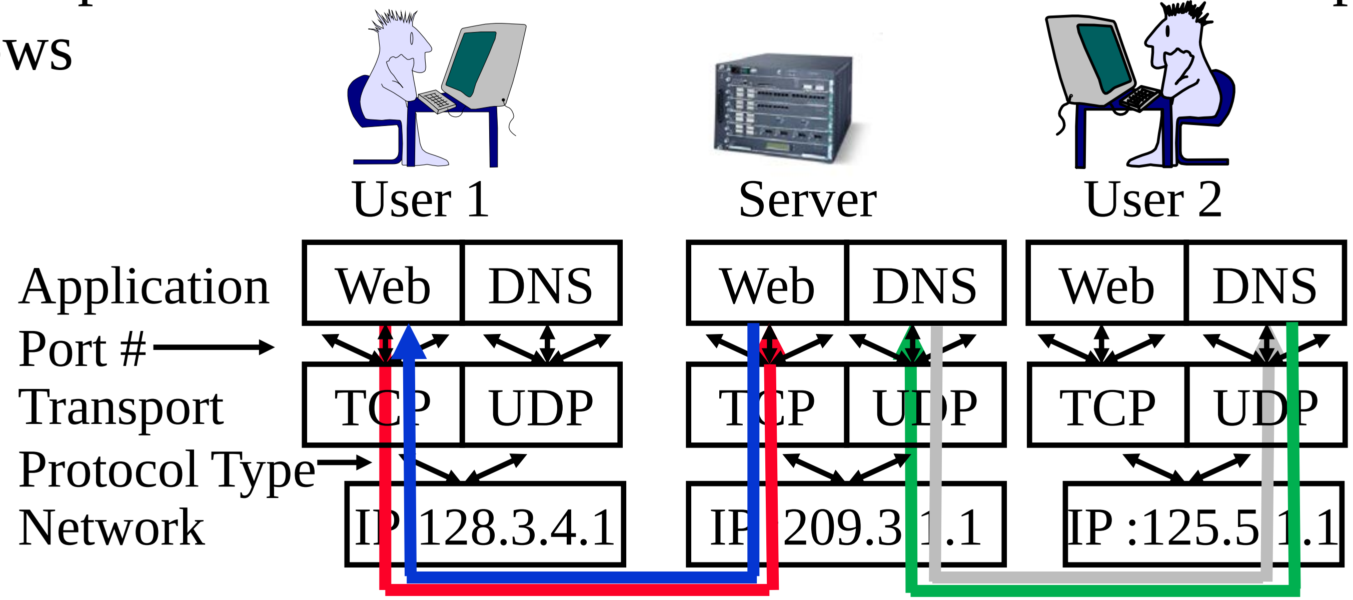
The source port numbers are randomly picked from a range outside the standard port #.

- If the sender does not specify a port number, which port does the transport layer resolve that message to?

The packet is dropped. Application s/w at the source puts in the default port # if needed.

Multiplexing and Demultiplexing

- Transport **Ports** and Network **addresses** are used to separate flows



HTTP Req.	SP:3009	DP:80	SA: 128.3.4.1	DA: 209.3.1.1	⇒
HTTP Resp.	SP:80	DP:3009	SA:209.3.1.1	DA:128.3.4.1	⇐
DNS Req.	SP:5009	DP:53	SA: 125.5.4.1	DA: 209.3.1.1	⇐
DNS Resp.	SP:53	DP:3009	SA:209.3.1.1	DA:125.5.4.1	⇒

Ref: http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

Student Questions

- We often use the default port number to communicate with the server. On a TCP or UDP-based server, wouldn't it cause any problems if many people were talking to the same port number?

Port numbers are like doors. Many people can arrive through the same door.

- Can a service port accept multiple requests within the same time frame? How is this traffic handled and manipulated by the port?

Multiple packets can enter the same port.

- Is the transport port the same as the application port?

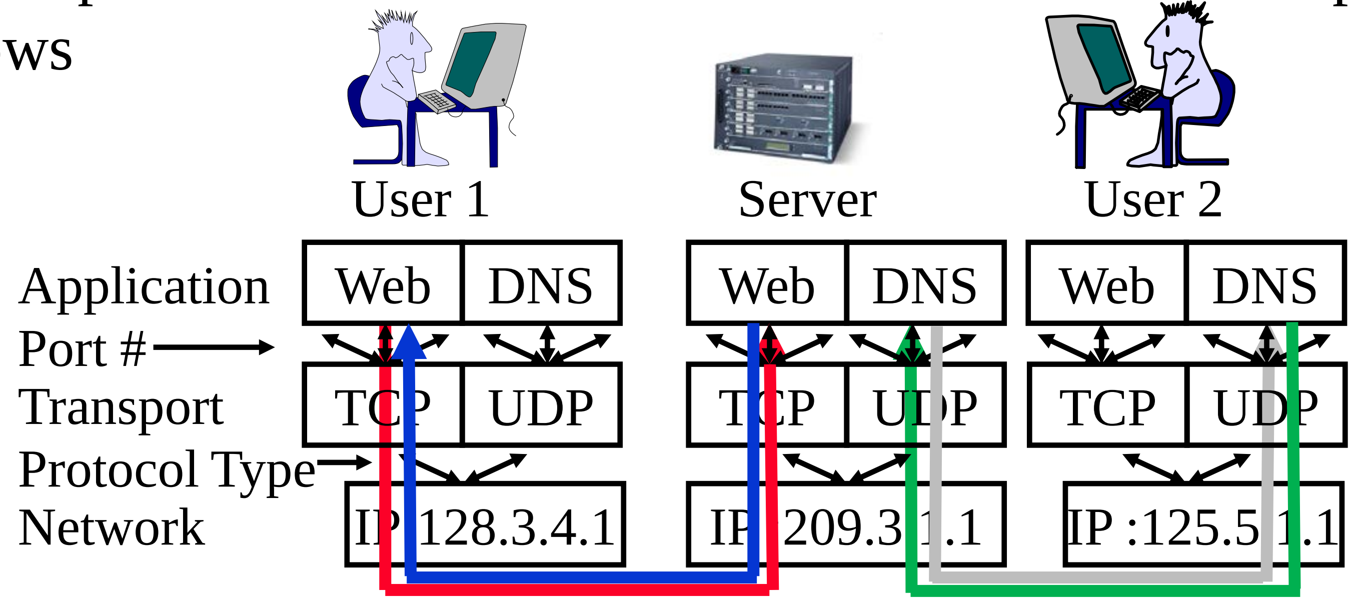
Yes. The transport port is the Service Access Point for transports

- Can you explain this diagram again? Can't see where the laser was pointing from the video. *Sure.*
- Can applications send and receive over the same port simultaneously? *Yes.*
- Can we cover slide 3-7 since I am having a hard time understanding the diagram?

Sure.

Multiplexing and Demultiplexing

- Transport **Ports** and Network **addresses** are used to separate flows



HTTP Req.	SP:3009	DP:80	SA: 128.3.4.1	DA: 209.3.1.1
HTTP Resp.	SP:80	DP:3009	SA:209.3.1.1	DA:128.3.4.1
DNS Req.	SP:5009	DP:53	SA: 125.5.4.1	DA: 209.3.1.1
DNS Resp.	SP:53	DP:3009	SA:209.3.1.1	DA:125.5.4.1

Ref: http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

Student Questions

- Is the multiplexing of ports usually done by frequency or time? *Multiplexing is on the link, not at the port. Packets from different applications use different port numbers to avoid confusion at the destination. After receiving, the packets are serialized in time, processed by lower layers, and queued by the port number for the application to process.*
- Is it possible for an application to use more than one port? If so, does it give them any benefit in terms of data transfer rates?

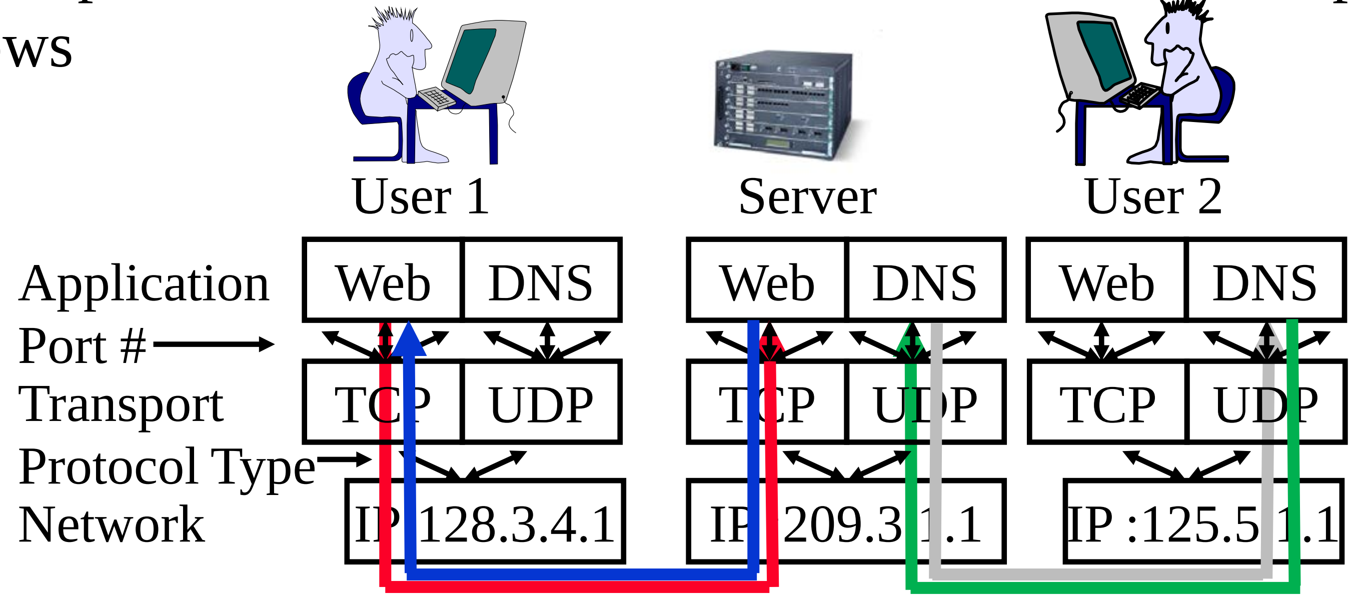
One application could be talking to many applications, each requiring a different port. For example, a movie player may talk to a video server, caption server, quiz server, and audio server. All 4 use different ports. It can also talk to two video servers – both using the same port but at different destinations.

- What are the advantages and disadvantages of multiplexing demultiplexing?

Multiplexing allows all clients to come through one port.

Multiplexing and Demultiplexing

- Transport **Ports** and Network **addresses** are used to separate flows



HTTP Req.	SP:3009	DP:80	SA: 128.3.4.1	DA: 209.3.1.1	⇨
HTTP Resp.	SP:80	DP:3009	SA:209.3.1.1	DA:128.3.4.1	⇦
DNS Req.	SP:5009	DP:53	SA: 125.5.4.1	DA: 209.3.1.1	⇦
DNS Resp.	SP:53	DP:3009	SA:209.3.1.1	DA:125.5.4.1	⇨

Ref: http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

Student Questions

- Are SA and DA also abbreviations for UDP and TCP port numbers? If not, then what are they?

SA=Source Address
DA=Destination Address
SP=Source Port
DP=Destination Port

- Is there any relation between this multiplexing and the multiplexing concept related to LED displays?

Not sure about LED multiplexing.

- What is a port specifically? Is it a hardware concept or more software?

Software concept. Was discussed extensively in Chapter 2.

- How does port forwarding work?

Port forwarding is related to the shortage of IP addresses and will be discussed in Chapter 4.

User Datagram Protocol (UDP)

- Connectionless end-to-end service
- Provides multiplexing via ports
- Error detection (Checksum) is optional. Applies to **pseudo-header** (same as TCP) and UDP segment. If not used, it is set to zero.
- No error recovery (no acks). No retransmissions.
- Used by network management, DNS, Streamed multimedia (Applications that are loss tolerant, delay-sensitive, or have their own reliability mechanisms)

Source Port	Dest Port	Length	Check-sum
16b	16b	16b	16b

← Size in bits

Student Questions

- Why would you use checksum with UDP since there is no way to recover the data if there is an error?

To throw away corrupted information.

- Can UDP be based on circuit switching? Circuit switching is required a complete connection in physics, but UDP is a connectionless end-to-end service. What is the difference between "connection" here?

Connection = Set up before talking.

- You mention RFC. What is that?

RFC = Request for Comments

RFCs started out as memos that were used by the initial Internet research team to inform other team members about their design decisions. These are numbered and are still used. Some of these are "informational," while others are "standards" that specify a protocol. TCP is specified in RFC 793:

https://tools.ietf.org/html/rfc793

- When do we take the pseudo header to the checksum? *Always for UPD and TCP.*
- Do most UDP transmissions opt to use the checksum or not? *Yes. It is almost universal. No apps can work with bad data.*

User Datagram Protocol (UDP)

- Connectionless end-to-end service
- Provides multiplexing via ports
- Error detection (Checksum) is optional. Applies to **pseudo-header** (same as TCP) and UDP segment. If not used, it is set to zero.
- No error recovery (no acks). No retransmissions.
- Used by network management, DNS, Streamed multimedia (Applications that are loss tolerant, delay-sensitive, or have their own reliability mechanisms)

Source Port	Dest Port	Length	Check-sum
16b	16b	16b	16b

← Size in bits

Student Questions

- Why use UDP over TCP even if you don't require reliability?

Reliability costs in terms of delay. In overloaded networks, packets may be lost, causing the receiver to wait for retransmissions. Sometimes it is better to move on and not wait for retransmissions. This is the case with audio/video.

- Those streaming multimedia applications are often considered not to be data loss sensitive. They may use UDP in the transport layer with respect to the textbook, but why do many of them are actually using TCP instead?

The application may have more than Video, such as file size (which is data). TCP is used for data.

- Why is UDP used instead of TCP？

Because there is no waiting for lost packet recovery.

- Why is this called "pseudo-header" and not just "header"?

It is prepared in the memory but not added to the packet.

User Datagram Protocol (UDP)

- Connectionless end-to-end service
- Provides multiplexing via ports
- Error detection (Checksum) is optional. Applies to **pseudo-header** (same as TCP) and UDP segment. If not used, it is set to zero.
- No error recovery (no acks). No retransmissions.
- Used by network management, DNS, Streamed multimedia (Applications that are loss tolerant, delay-sensitive, or have their own reliability mechanisms)

Source Port	Dest Port	Length	Check-sum	
16b	16b	16b	16b	← Size in bits

Student Questions

- Is the purpose of a pseudo-header only to carry the checksum? *Yes*
- Does TCP also use a pseudo-header, or is it only used by UDP? *Both*
- Can an application use TCP and UDP at the same time? *Yes*
- UDP has no congestion control. How can it be applied in multimedia when people all use multimedia at the same time?

They all lose some packets. Recently congestion control has been added to UDP. But that is part of the "Recent Advances in Networking Course."

- Why is DNS loss tolerant?

If lost, DNS can try again.

- How does the transport layer assign multiple ports not being used for UDP?

A port is bound to only one process. Once used, the port can not be used for another process.

User Datagram Protocol (UDP)

- Connectionless end-to-end service
- Provides multiplexing via ports
- Error detection (Checksum) is optional. Applies to **pseudo-header** (same as TCP) and UDP segment. If not used, it is set to zero.
- No error recovery (no acks). No retransmissions.
- Used by network management, DNS, Streamed multimedia (Applications that are loss tolerant, delay-sensitive, or have their own reliability mechanisms)

Source Port	Dest Port	Length	Check-sum	
16b	16b	16b	16b	← Size in bits

Student Questions

- Do any UDP application implement their own packet sequence number?

Yes. Several IoT apps do not care for lost packets but want to use only newer information.

- Why does DNS use UDP?

Since it is a simple request-response, TCP connection overhead cannot be justified.

- What makes UDP loss tolerant?

It does not take care of losses.

- Are there any other error detection methods except checksum?

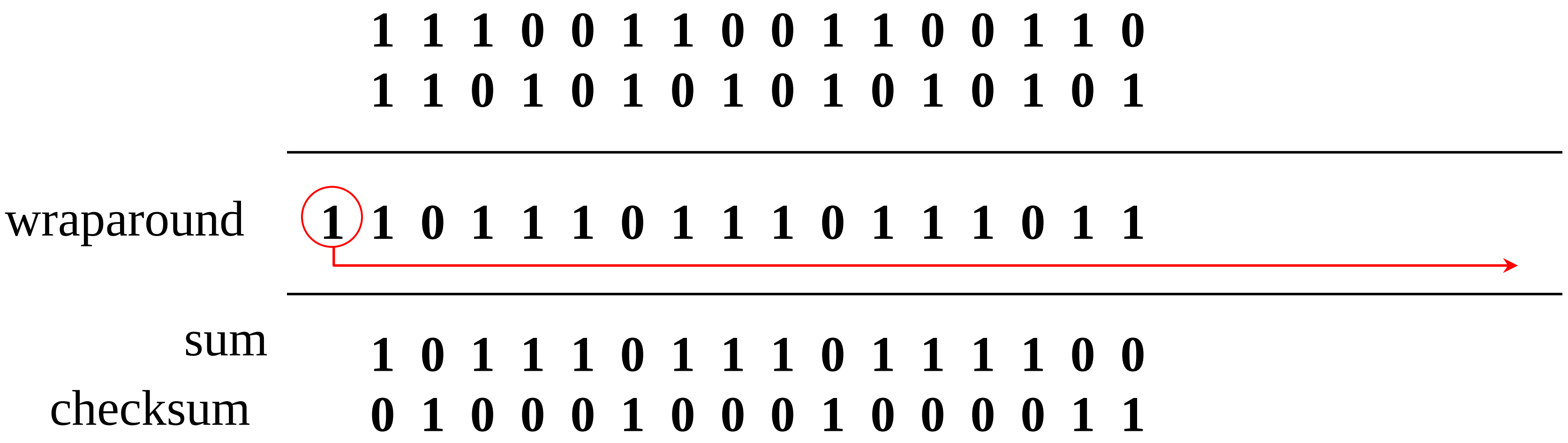
Yes. We will see them in Layer 2.

- How heavily do checksums affect the reliability and performance of UDP-based applications, particularly in environments where network conditions vary widely?

Checksum is optional in UDP. Generally, the applications are designed to be less tolerant. If the loss rate becomes high, the application stops.

Error Detection: Checksum

- **Cyclic Redundancy Check (CRC)**: Powerful but generally requires hardware
- **Checksum**: Weak but easily done in software
  - **Example**: *1's complement* of 1’s complement sum of 16-bit words with overflow wrapped around



```
               1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
               1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
            ------------------------------------
wraparound   1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1
            ------------------------------------
       sum     1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0
  checksum     0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1
```

At the receiver, the sum is all 1’s, and the checksum is zero.

Student Questions

- Since in 2’s complement, the smallest number (the largest negative number) has a greater magnitude than the most positive on fixed-sized ALU's, how do you negate that number since there is no positive equivalent?

We don’t use that number.

- Besides checksum, any other ways for CRC?

Many other ways. More in Chapter 6.

- Are the two 16-bit words predefined by the application? Trying to understand the purpose checksum provides.

This is just an example. All bits in the packets are arranged as rows of 16 bits. Then the checksum is computed and added to the header.

- What is an error that can arise from getting -0

In all cases, -0 is considered the same as 0. So there is no error.

- Is 1's compliment better than 2’s?

No. But, some tricks work with one representation but not the other, and so they are used in a different context.

Error Detection: Checksum

- ❑ **Cyclic Redundancy Check (CRC)**: Powerful but generally requires hardware
- ❑ **Checksum**: Weak but easily done in software
  - ➢ **Example**: *1's complement* of 1’s complement sum of 16-bit words with overflow wrapped around

```
              1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
              1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
           ------------------------------------
wraparound  1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1
           ------------------------------------
sum           1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0
checksum      0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1
```

At the receiver, the sum is all 1’s, and the checksum is zero.

Student Questions

- ❑ Why is it that we take the 1's complement sum of the header and message for checksum instead of just checking if the two checksums are equal?

At the receiver, the checksum comes out zero. Rather than comparing if it is 25, you can add -25 to the message so that the receiver will get 0.

- ❑ If the checksum(1's complement) becomes wrong when sent to the receiver, will it still be wrong even if the other words are all correct?

Yes. We do know which byte is in error. The entire message will be considered in error.

- ❑ Can you please explain what " 1's complement of 1’s complement sum" on slide 3-9 means? Isn't 1's complement of 1’s complement sum just the sum itself?

No. All operations here are done using “1’s complement arithmetic.”
Please do not confuse the name of the method with the operation.
Checksum = Complement(Sum)
You are thinking
Checksum = Complement(Complement(Sum))

Error Detection: Checksum

- **Cyclic Redundancy Check (CRC)**: Powerful but generally requires hardware
- **Checksum**: Weak but easily done in software
  - **Example**: *1's complement* of 1's complement sum of 16-bit words with overflow wrapped around

```
                1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
                1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
              ---------------------------------
wraparound    1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1
              └─────────────────────────────→ 1
              ---------------------------------
sum             1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0
checksum        0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1
```

At the receiver, the sum is all 1's, and the checksum is zero.

Student Questions

- Book pp. 199 - 3.3.2 UDP checksum. This question applies to TCP and UDP checksums. If we have two bytes: 0000 0000 and 0000 1111, and they are corrupted such that the new (corrupted) bytes are 0000 0010 and 0000 1101, won't they still pass the checksum? The probability of this is quite low, but it could happen. Does TCP have any recourse for this?

This is the case of a two-bit error. Two-bit errors may or may not be detected by the TCP checksum. If not detected, the message will be considered correct and delivered to the application unless other inconsistencies prevent it from being delivered.

- Can we go through an example of calculating TCP checksum?

Sure.

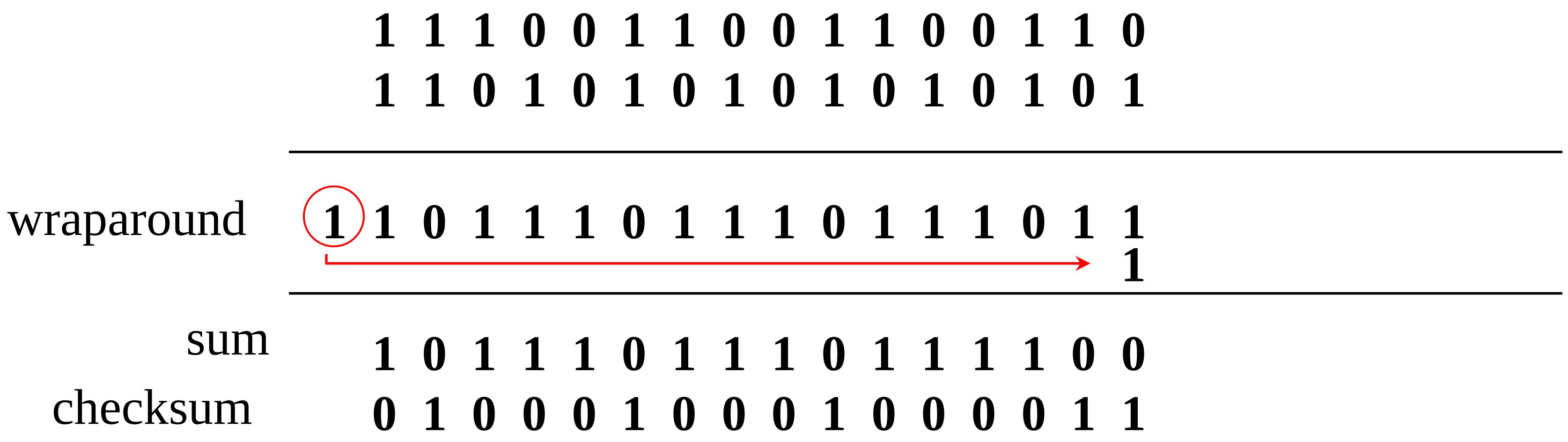
- Does UDP throw the entire packet? Or does only the part has errors?

- *We don't know which part has an error. The entire packet is dropped.*

- Why is checksum weak? *Does not detect many errors.*

Error Detection: Checksum

- **Cyclic Redundancy Check (CRC)**: Powerful but generally requires hardware
- **Checksum**: Weak but easily done in software
  - **Example**: *1's complement* of 1’s complement sum of 16-bit words with overflow wrapped around



```
                1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
                1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
            ---------------------------------------
wraparound    1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1
                                                1
            ---------------------------------------
       sum      1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0
  checksum      0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1
```

At the receiver, the sum is all 1’s, and the checksum is zero.

Student Questions

- If something is wrong with both the data bits and checksum bits, is that possible that the wrong checksum could just match the wrong data, and we will not detect the error in data?

There is a small probability of undetected errors.

- What are the fields being summed? Is it every 16-bit word? *Yes, every 16-bit word.*
- Why do we drop the 1 in the front?

It is wrapped around.

- Can the checksum field be corrupted as well? *Yes. All bits, including checksum, are covered.*
- What hardware is needed for CRC?

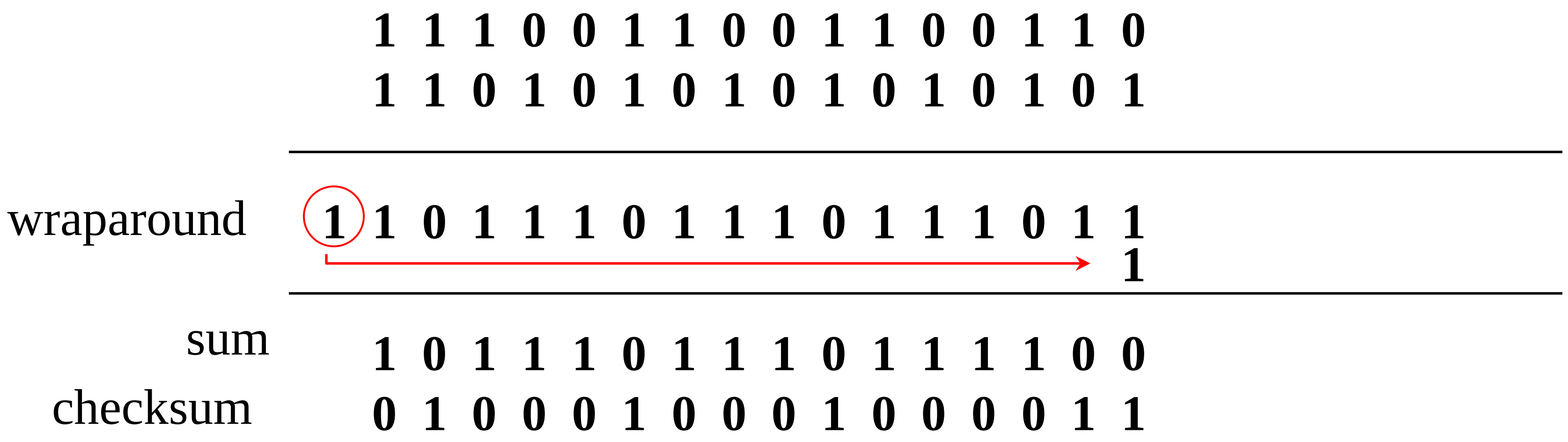
Shift-registers

- What’s the difference between the checksum and Hamming code?

- *Hamming code can correct some errors. Checksum cannot correct bit errors.*

Error Detection: Checksum

- ❑ **Cyclic Redundancy Check (CRC)**: Powerful but generally requires hardware
- ❑ **Checksum**: Weak but easily done in software
  - ➢ **Example**: *1's complement* of 1’s complement sum of 16-bit words with overflow wrapped around



```
              1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
              1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
            ----------------------------------
wraparound  1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1
                                              1
            ----------------------------------
       sum    1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0
  checksum    0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1
```

At the receiver, the sum is all 1’s, and the checksum is zero.

Student Questions

- ❑ Can the data be changed so the checksum remains the same?

Yes, checksum does not detect many errors.

- ❑ So CRC is preferred, but our limitations do not make it viable much?

No. IP was designed so that it can be implemented all in software. Ethernet requires hardware and uses CRC.

- ❖ Could you do another example, e.g., Book p. 199?

Yes. See Slide 3.73 for four words.

1’s Complement

2’s Complement: -ve of a number is complement+1

- 1 = 0001 -1 = 1111
- 2 = 0010 -2 = 1110
- 0 = 0000 -0 = 0000

1’s complement: -ve of a number is its complement

- 1 = 0001 -1 = 1110
- 2 = 0010 -2 = 1101
- 0 = 0000 -0 = 1111

2’s Complement sum: Add with carry. Drop the final carry, if any.

6-7 = 0110 + (-0111) = 0110 + 1001 = 1111 => -1

1’s complement sum: Add with carry. Add end-around carry back to sum

- 6-7 = 0110 + (-0111) = 0110+1000 = 1110 => -1

Complement of 1’s complement sum: 0001

Checksum: At the transmitter: 0110 1000, append 0001

At the receiver: 0110 1000 0001 compute checksum of the full packet
= complement of sum = complement of 1111 = 0000

Ref: https://en.wikipedia.org/wiki/Ones%27_complement

Student Questions

- If we are sending 16-bit packets and we want to use a checksum, how many bits of the packet are the actual data, and how many bits of the packet is the complement sum?

16-bit packets would be too small for anything. A packet consists of many 16-bit words. One extra 16-bit word is added as a checksum.

- Does 2's complement relate to sign bit in this case?

No. Two’s complement relates to the entire word, not just the sign bit.

- For the example of checksum, how can we get the appended number(0001 in this case) at the transmitter?

By computing the complement of the 2’s complement sum of all words in the packet.

- For the scope of this exam, when would we need to use 2's compliment?

2’s complement is presented here to contrast with 1’s complement. We may use 2’s complement during the CRC discussion in Chapter 6.

1’s Complement

2’s Complement: -ve of a number is complement+1

- 1 = 0001 -1 = 1111
- 2 = 0010 -2 = 1110
- 0 = 0000 -0 = 0000

1’s complement: -ve of a number is its complement

- 1 = 0001 -1 = 1110
- 2 = 0010 -2 = 1101
- 0 = 0000 -0 = 1111

2’s Complement sum: Add with carry. Drop the final carry, if any.

6-7 = 0110 + (-0111) = 0110 + 1001 = 1111 => -1

1’s complement sum: Add with carry. Add end-around carry back to sum

- 6-7 = 0110 + (-0111) = 0110+1000 = 1110 => -1

Complement of 1’s complement sum: 0001

Checksum: At the transmitter: 0110 1000, append 0001

At the receiver: 0110 1000 0001 compute checksum of the full packet
= complement of sum = complement of 1111 = 0000

Ref: https://en.wikipedia.org/wiki/Ones%27_complement

Student Questions

- 1's complement addition almost seems pointless since it's so easy to convert two binary numbers to base 10, normally add, then convert back to 1's complement. Is there any reason to avoid doing it this way?

Yes. Computers do not know decimal arithmetic. They only know binary arithmetic.

- How are checksum and parity bit error detection different? *Parity is a single bit to protect a single word. The checksum is a word to protect multiple words.*
- Where does 1111 in the last line come from? *It’s the 1’s complement sum of the 3 words in the message: 0110 1000 0001*
- Is the checksum computed at every hop or just at the destination?

TCP is only at the destination

- Can you add these examples you did on the board and annotate the things you are pointing to in the lecture slides?

Generally, these are the things on the slide written differently.

1’s Complement

2’s Complement: -ve of a number is complement+1

- 1 = 0001 -1 = 1111
- 2 = 0010 -2 = 1110
- 0 = 0000 -0 = 0000

1’s complement: -ve of a number is its complement

- 1 = 0001 -1 = 1110
- 2 = 0010 -2 = 1101
- 0 = 0000 -0 = 1111

2’s Complement sum: Add with carry. Drop the final carry, if any.

6-7 = 0110 + (-0111) = 0110 + 1001 = 1111 => -1

1’s complement sum: Add with carry. Add end-around carry back to sum

- 6-7 = 0110 + (-0111) = 0110+1000 = 1110 => -1

Complement of 1’s complement sum: 0001

Checksum: At the transmitter: 0110 1000, append 0001

At the receiver: 0110 1000 0001 compute checksum of the full packet
= complement of sum = complement of 1111 = 0000

Ref: https://en.wikipedia.org/wiki/Ones%27_complement

Student Questions

- Can we go over this again during the Q&A?

Sure

- Why do we send the complement of 1's complement at the end of the checksum

That’s the procedure

- Can you go over how the checksum works again?

Sure.

- Can you explain 1’s complement vs 2’s complement?

Sure.

- What does "add with carry" mean? And what does "end-around carry" mean?

Add w carry = Take carry to the next adjacent position.

End-around carry = Take carry back to the first position.

Homework 3A: Checksum

[6 points] Consider the following two 16-bit words: ABCD 1234

A. What is the checksum as computed by the sender

B. Add your answer of Part A to the end of the packet and show how the receiver will compute the checksum of the received three 16-bit words and confirm that there are no errors.

C. Now assume that the first bit of the packet is flipped due to an error. Repeat Part B at the receiver. Is the error detected?

Student Questions

- ❑ In Homework3A, why do we use 1's Complement? Why not 2's Complement?

The homework does not specify which complement to use. Please use the complement that will be used by UDP/TCP.

- ❑ For the encoding of ABCD, would it just be decimal values 10-14 in binary? Assuming 0-9 are the same in their binary encoding.

$9_{16} = 9_{10} = 1001_2$

$A_{16} = 1010_2$

$19_{10} = 13_{16} = 00010011_2$

- ❑ Is the first bit of the packet referring to the entire packet, including headers? Or first bit of a message?

We have not used "message" in this homework. The packet consists of two words indicated plus one word of checksum. Total three words.

Homework 3A: Checksum

[6 points] Consider the following two 16-bit words: ABCD 1234

A. What is the checksum as computed by the sender

B. Add your answer of Part A to the end of the packet and show how the receiver will compute the checksum of the received three 16-bit words and confirm that there are no errors.

C. Now assume that the first bit of the packet is flipped due to an error. Repeat Part B at the receiver. Is the error detected?

Student Questions

- Book Page 211: On the receiver side, how to compute the checksum of the acknowledgment packet being sent to layer 3?

As usual, the ack packet will be assembled, and the checksum computed at the transmitter and checked at the receiver. Nothing special about ack packets. Most acks come with data as data packets anyway.

UDP: Summary

1. UDP provides flow multiplexing using port #s
2. UDP optionally provides error detection using the checksum
3. UDP does not have an error or loss recovery mechanism

Read Sections 3.1, 3.2, and 3.3, Do R1-R8, P2-P5

Student Questions

❑ How can UDP provide fragmentation if it does not support packet sequencing?

IP does fragmentation and reassembly for all packets, even those from UDP. The max UPD packet size is 2^{16}-1 word since the length field is 16 bits.

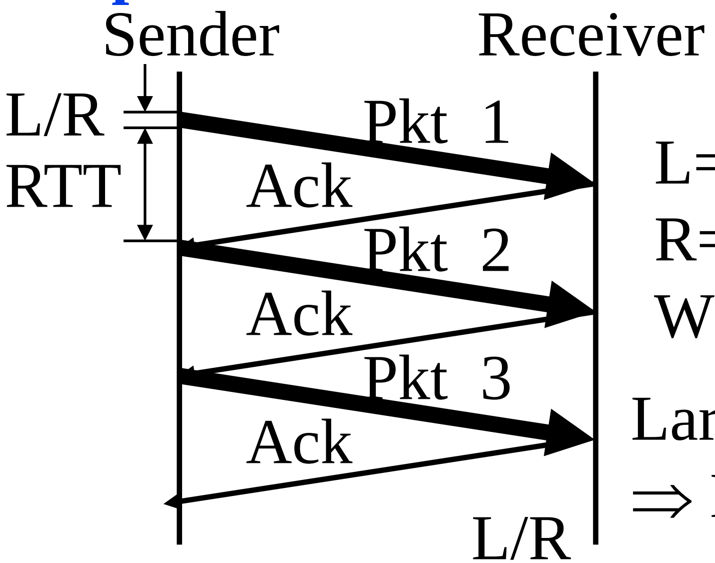
❖ Chapter 3.3, page 197, figure 3.6: what is the NFS protocol, and why is it using UDP for "remote server?“

Network File Systems designer wanted to keep the server stateless. The receiver could request missing blocks.

Flow Control

- Flow Control Goals:
  1. Sender does not flood the receiver,
  2. Maximize throughput

Stop and Wait Flow Control



$$\text{Throughput} = \frac{L/R}{RTT+L/R}$$

L= Packet Length

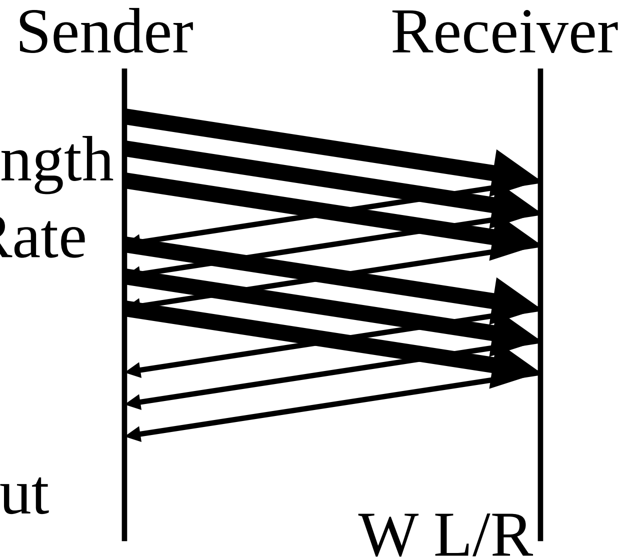
R= Link bit Rate

W= Window

Large RTT

⇒ Low Thruput

Window Flow Control



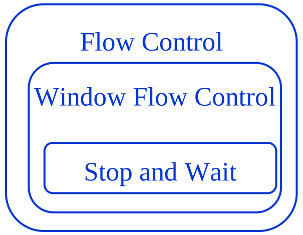
$$\text{Throughput} = \frac{W\,L/R}{RTT+L/R}$$

Ref: Textbook Section 3.4.2

Student Questions

- What's the actual difference between Stop and Wait and Window Flow Control

The following Venn diagram explains that Stop and Wait is one of the of flow control methods.



- Why, in this case, (L/R)/(RTT=L/R) is the throughput? I mean, according to slide 15 is also called utilization.

Utilization is between 0 and 1. Throughput is in bits/second.

- The unit of the throughout discussed before is bps. Why does the throughput become a ratio now?

Ratio=% throughput

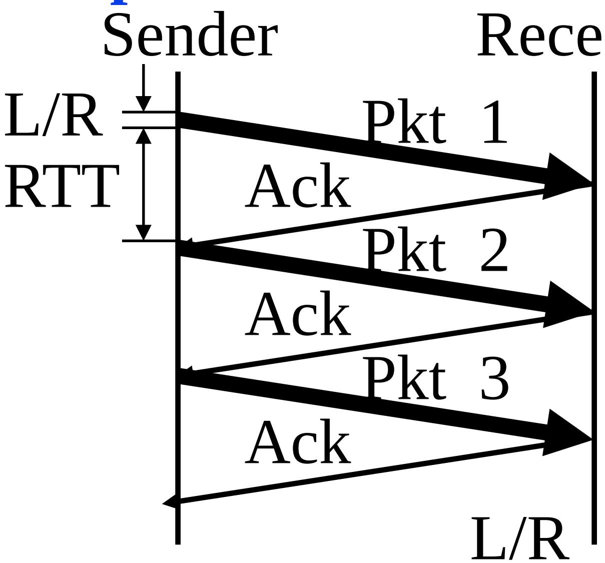
- Should WL/R be smaller than RTT+L/R?

Not necessarily. However, this relative throughput cannot be more than 1.

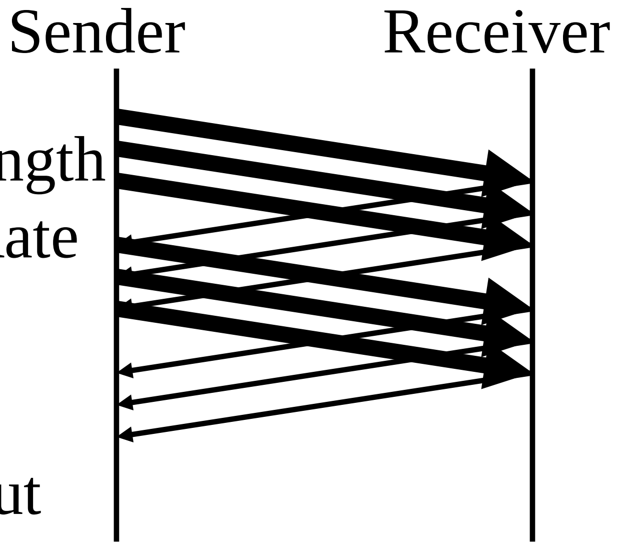
Flow Control

- Flow Control Goals:
  1. Sender does not flood the receiver,
  2. Maximize throughput

Stop and Wait Flow Control



Window Flow Control



L= Packet Length

R= Link bit Rate

W= Window

Large RTT

⇒ Low Thruput

$$\text{Throughput} = \frac{L/R}{RTT+L/R}$$

$$\text{Throughput} = \text{Min}\{\frac{WL/R}{RTT+L/R},1\}$$

Ref: Textbook Section 3.4.2

Student Questions

- In the textbook, the GBN and SR protocol is attributed to the reliable transmission service. But the class attributed them to the flow control technology. Does flow control have to use techniques for a reliable data transfer service?

Any violation of flow control will result in a packet loss which is the same as unreliability. Bit errors also result in unreliability but will not violate SR or GBN. So flow control is a more correct classification for SR or GBN. This is just a necessary condition for reliable transmissions but not sufficient.

- Are there other types of flow control that are commonly used?

Many more to come in this module itself.

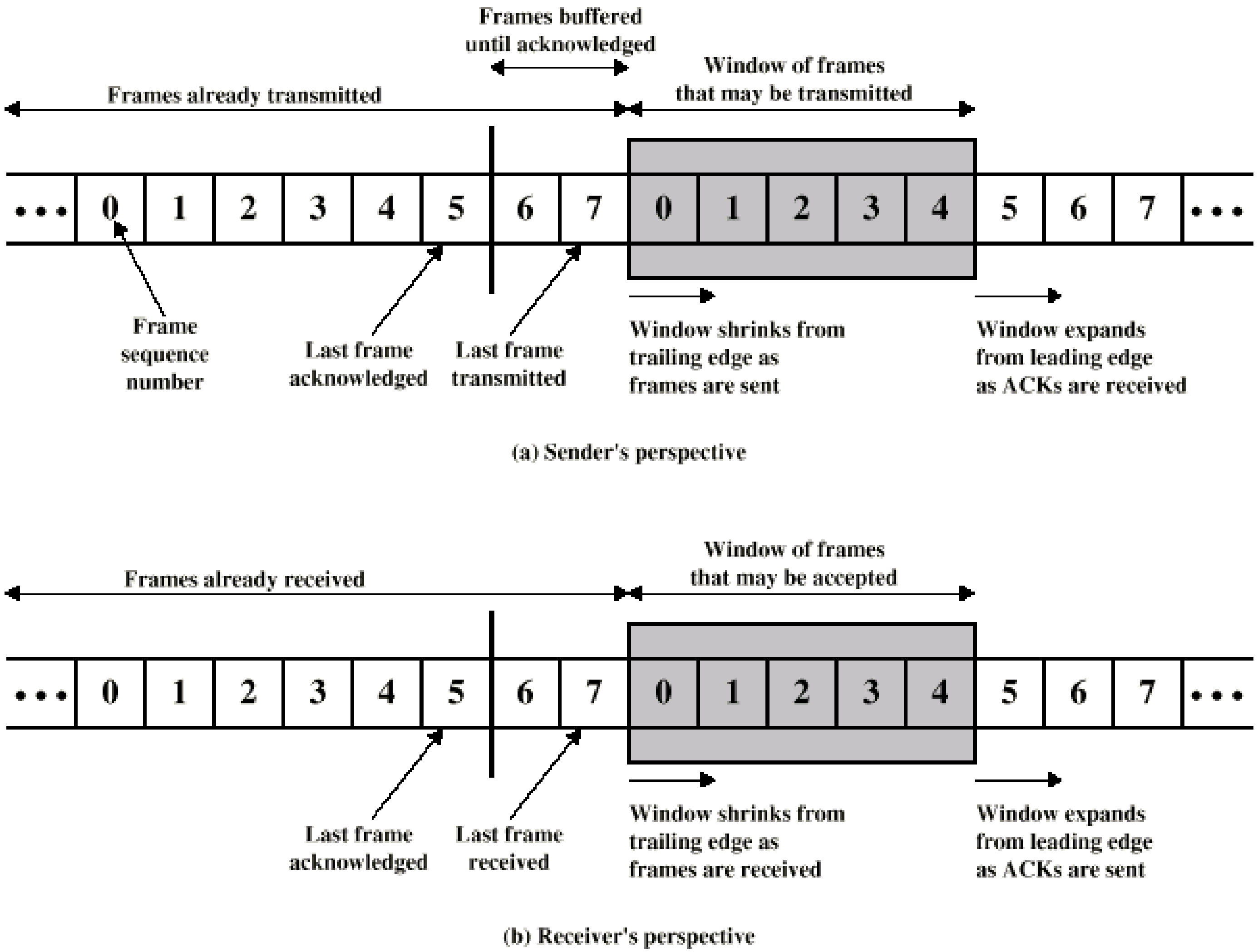
- Is throughput used the same as utilization?

Users want high throughput (goodput). System owners want high utilization. Both go together. However, sometimes utilization is high and throughput is low because of problems.

- In the flow control diagrams, the server seems to always ACK for the sequence number sent (i.e. , client sends 1 and server ACKs 1), but in the three-way handshake, the server ACKs to client is n + 1, and the client ACKs to server is n + 1. Why is that?

Handshake is TCP. This slide is not TCP.

Sliding Window Diagram


Frames buffered until acknowledged
Window of frames that may be transmitted
Frames already transmitted
• • • 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 • • •
Frame sequence number
Last frame acknowledged
Last frame transmitted
Window shrinks from trailing edge as frames are sent
Window expands from leading edge as ACKs are received
Window of frames that may be accepted
Frames already received
• • • 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 • • •
Last frame acknowledged
Last frame received
Window shrinks from trailing edge as frames are received
Window expands from leading edge as ACKs are sent

(a) Sender's perspective

(b) Receiver's perspective

Student Questions

- ❑ Is there a maximum size for the window? If 6 and 7 get acked, will the sender's window expand by 2 frames, or will it wait till it sends 2 frames before expanding?

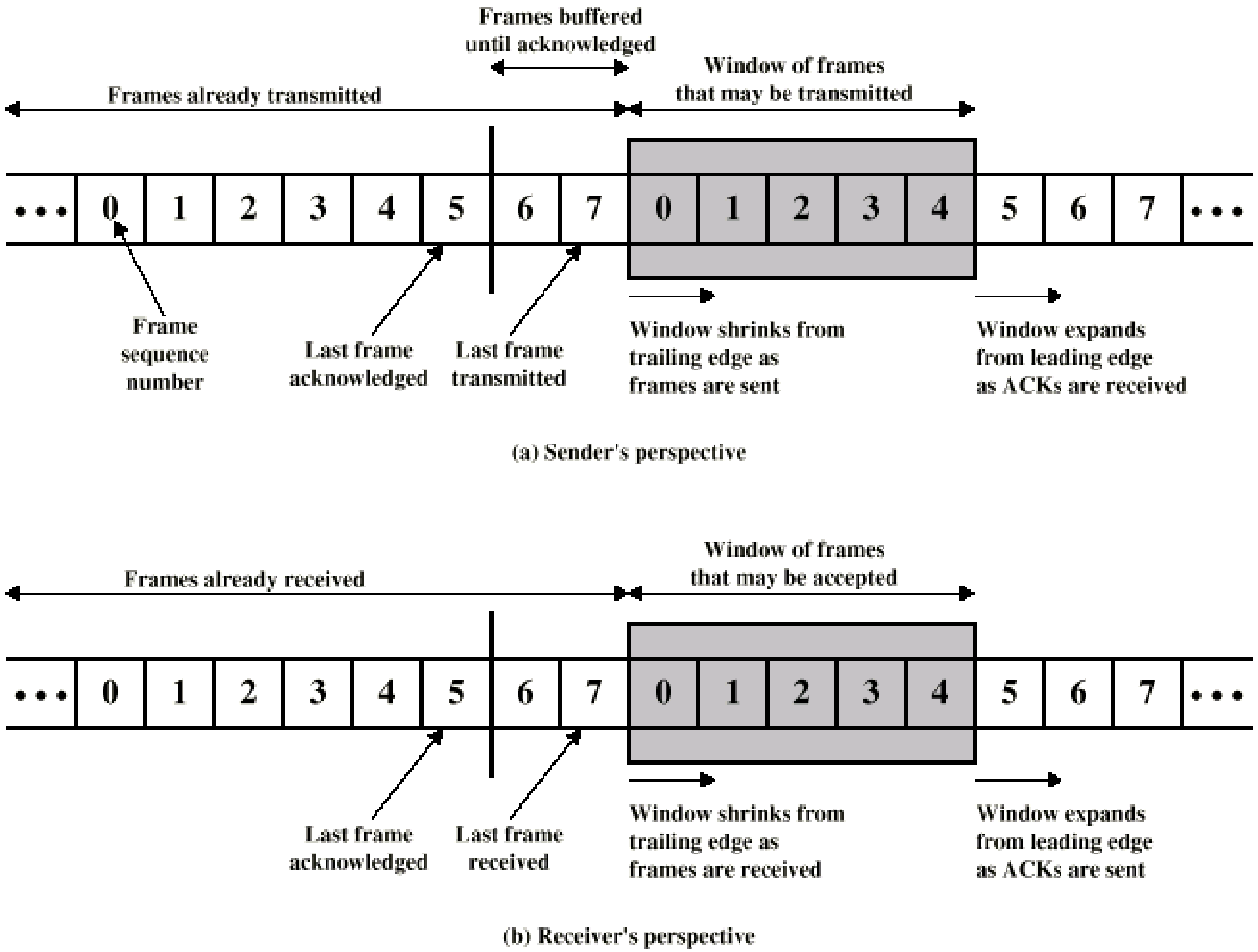
*The window shows the packets that can be sent **but have not been sent**. The maximum size is determined by the receiver. The acks generally include instructions for how many more packets the receiver can receive. This slide shows a **special case** in which each ack that is acking n packets allows n more packets. This is not necessary.*

- ❑ Could you explain the sliding window diagram again since we can't see the pointer on the video? *Sure.*
- ❑ Is the size of the window fixed? Is the number of unacknowledged packets limited by a certain threshold?

Yes. That is called "window size."

- ❑ Is the size of the window fixed during transmission? *The receiver can change it anytime. This example shows a fixed size.*

Sliding Window Diagram



Frames buffered until acknowledged
Frames already transmitted
Window of frames that may be transmitted
• • • 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 • • •
Frame sequence number
Last frame acknowledged
Last frame transmitted
Window shrinks from trailing edge as frames are sent
Window expands from leading edge as ACKs are received
(a) Sender's perspective
Frames already received
Window of frames that may be accepted
• • • 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 • • •
Last frame acknowledged
Last frame received
Window shrinks from trailing edge as frames are received
Window expands from leading edge as ACKs are sent
(b) Receiver's perspective

Student Questions

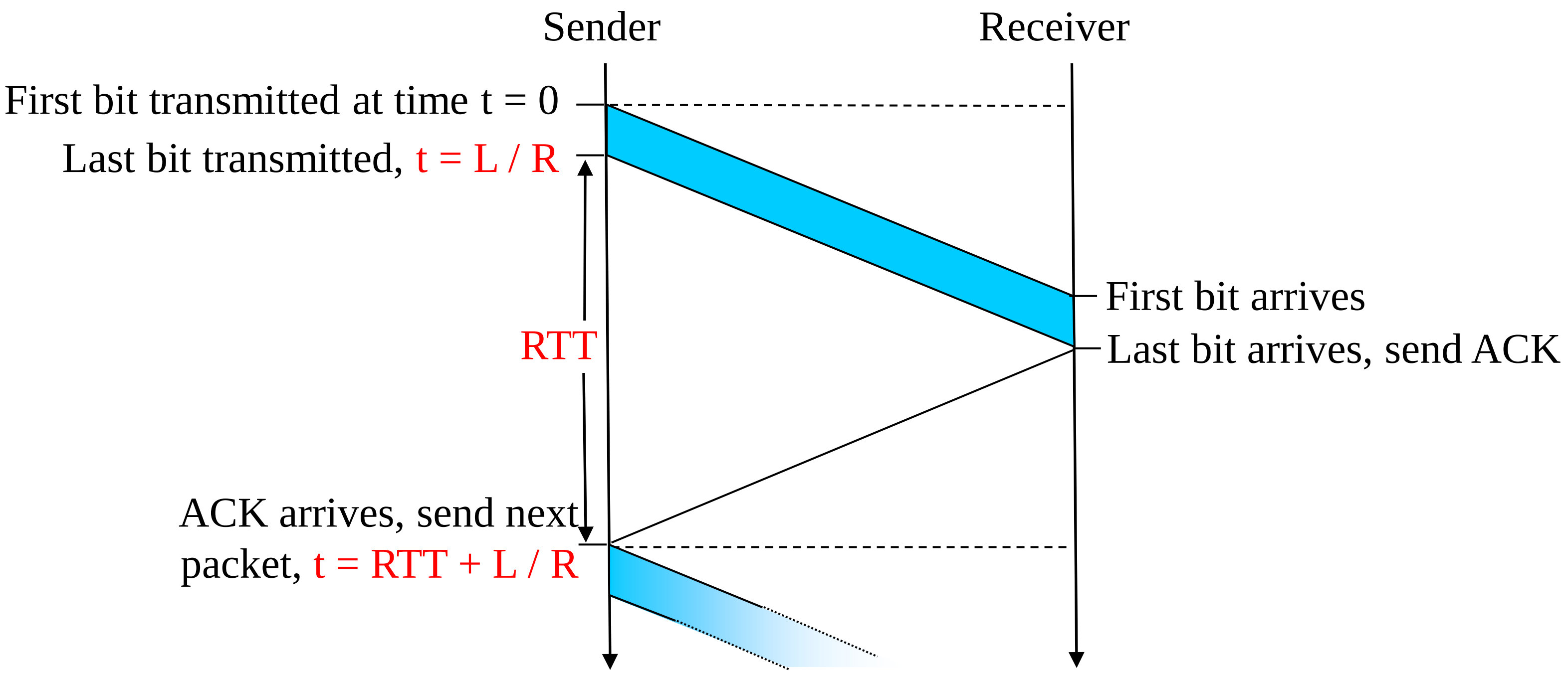
- ❑ Are the windows sizes communicated between the client and server? Is this part of the header?

The receiver communicates it to the sender.
Yes, this is communicated in the TCP header.

- ❑ Is window size N the same at both sender and receiver? Is this established during the handshake?

Yes, it is same. It is exchanged with every few packets.

Stop and Wait Flow Control



Sender
Receiver
First bit transmitted at time t = 0
Last bit transmitted, t = L / R
RTT
First bit arrives
Last bit arrives, send ACK
ACK arrives, send next packet, t = RTT + L / R

$$\text{Utilization } U = \frac{L / R}{RTT + L / R} = \frac{t_{frame}}{2t_{prop}+t_{frame}} = \frac{1}{2\alpha + 1}$$

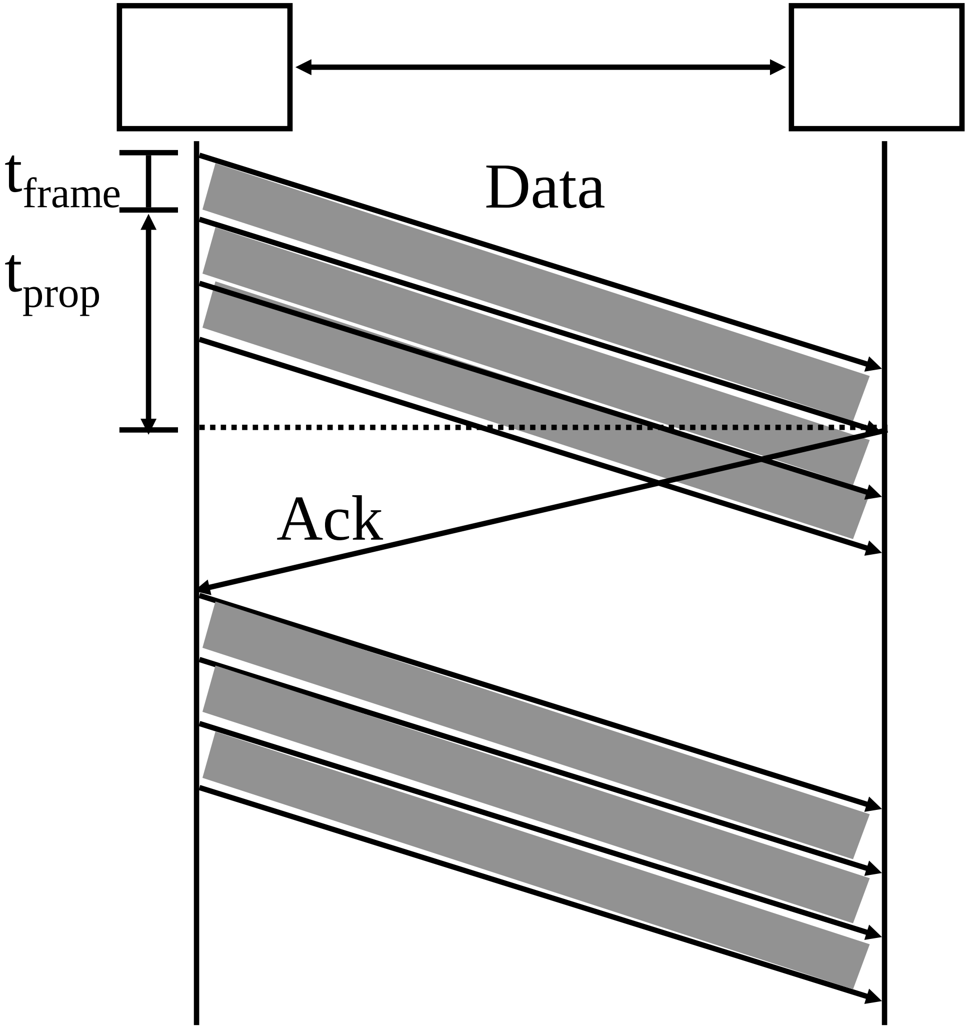
$$\text{Here, } \alpha = t_{prop}/t_{frame}$$

Student Questions

- ❑ Why is the round trip time equal to $2 \cdot t_{prop}$? Is this include processing and queuing delays?

t_{prop} here is actually a one-way delay. It includes propagation, processing, and queueing.

Sliding Window Protocol Efficiency



$$U = \frac{W\, t_{frame}}{2t_{prop} + t_{frame}}$$

$$= \begin{cases} \dfrac{W}{2\alpha+1} \\ 1 \text{ if } W > 2\alpha+1 \end{cases}$$

Here, $\alpha = t_{prop}/t_{frame}$

W=1 ⇒ Stop and Wait

Student Questions

Utilization: Examples

Satellite Link: One-way Propagation Delay = 270 ms
RTT=540 ms
Frame Size L = 500 Bytes = 4 kb
Data rate R = 56 kbps ⇒ t_{frame} = L/R= 4kb/56kbps =0.071s = 71 ms
$\alpha = t_{prop}/t_{frame} = 270/71 = 3.8$
$U = 1/(2\alpha+1) = 0.12$

- ❑ Short Link: 1 km = 5 μs (Assuming Fiber 200 m/μs),
  Rate=10 Mbps,
  Frame=500 bytes ⇒ t_{frame}= 4k/10M= 400 μs
  $\alpha=t_{prop}/t_{frame}=5/400=0.012 \Rightarrow U=1/(2\alpha+1)=0.98$

Note: The textbook uses RTT in place of t_{prop} and L/R for t_{frame}

Student Questions

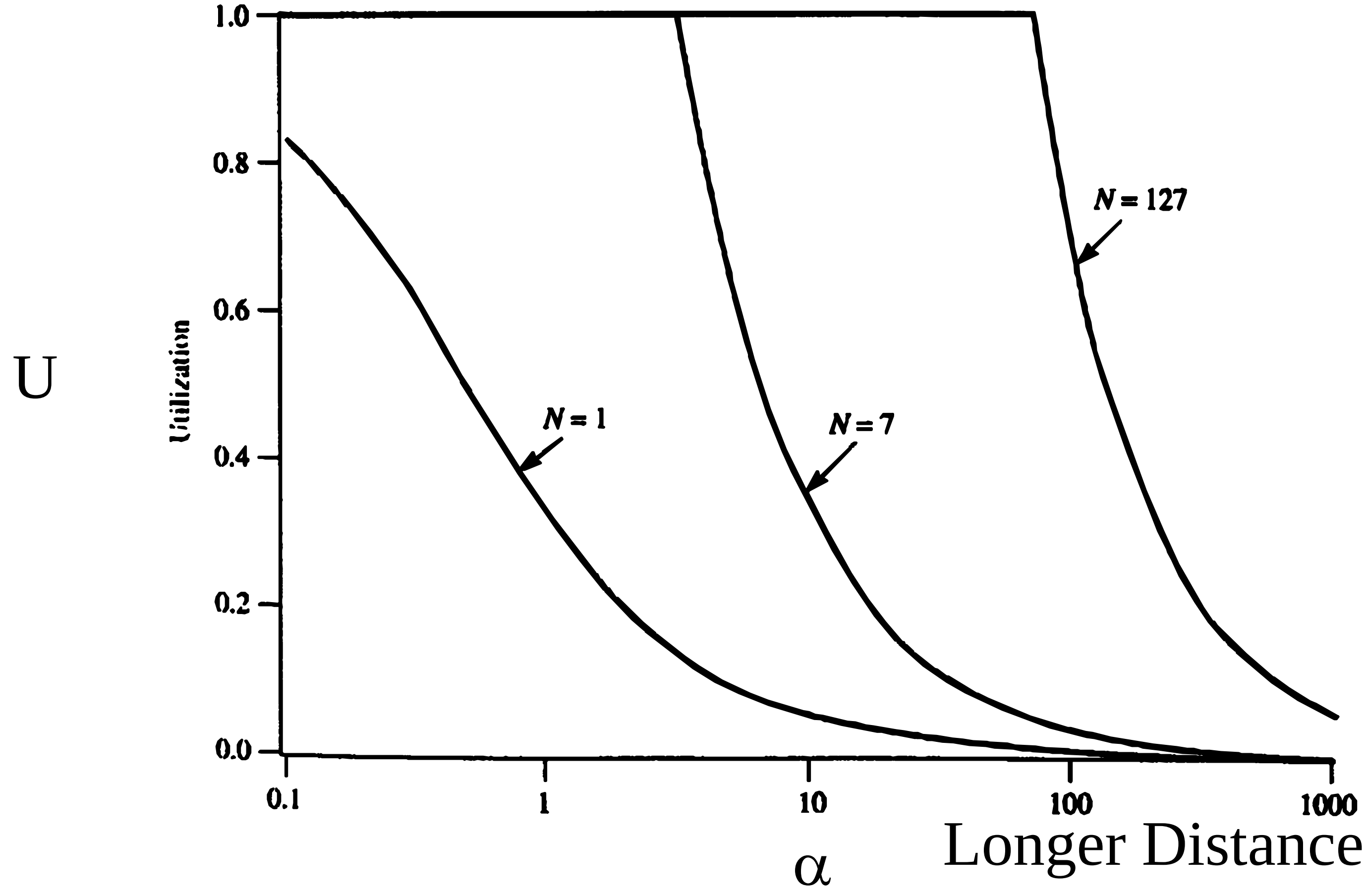
- ❑ When calculating U, should we use RTT or one-way propagation delay?

α uses one-way propagation delay

- ❑ If k represents 10^3 and K represents 1024, why M is 10^6 instead of something 2 based?

Storage people use 2-based, and they use bytes. Networking uses bits and powers of 10.

Effect of Window Size



Longer Distance →

Higher Bit Rate →

- Larger window is better for larger α

Student Questions

- Why is a larger window better for a larger α?

The graph shows that the larger N gives better utilization for larger α*.*

Large α
⇒ Long-distance network

For full utilization, you must fill up the entire path with packets, so you will need more packets as the distance increases. If the path is empty, the throughput is low. If the path is overfilled, the delay increases due to queueing.

- What downside is there to increasing window size?

Need more buffers at the receiver, routers, and sender.

- What is the disadvantage of using a large N value for small alpha values?

Need more buffers at the receiver, routers, and sender.

- Since the speed factor is in the denominator. Wouldn't a lower speed increase alpha, necessitating a larger sliding window?

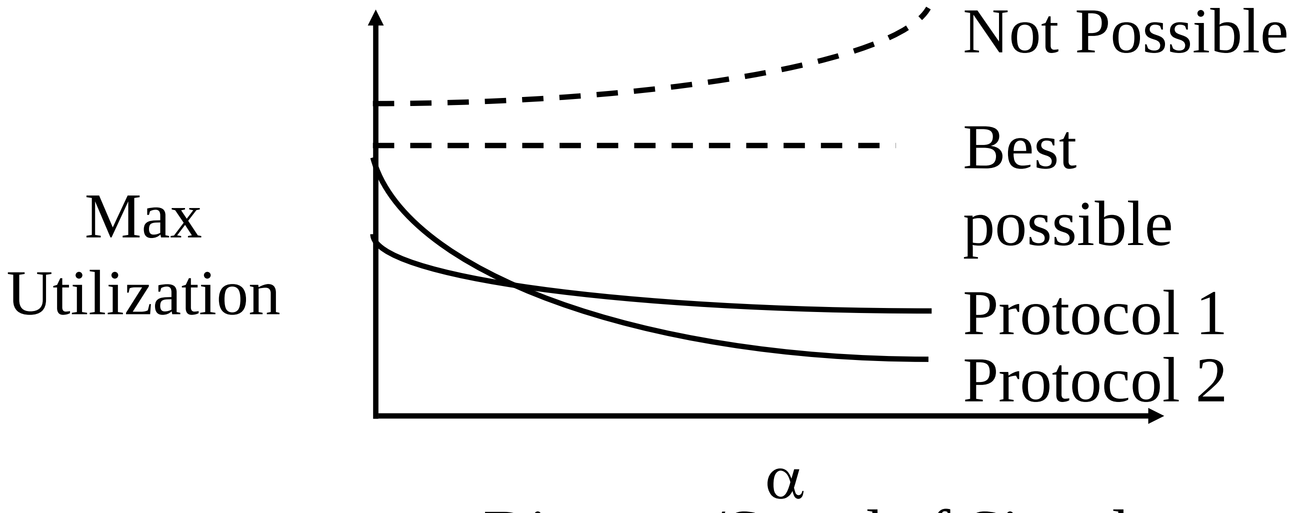
Speed was meant to indicate bit rate, not signal speed.

- How does window size affect network utilization?

See Slide 3.24

Efficiency Principle

- For **all** protocols, the maximum utilization (efficiency) is a *non-increasing* function of α.



$$\alpha = \frac{t_{prop}}{t_{frame}} = \frac{\text{Distance/Speed of Signal}}{\text{Bits Transmitted /Bit rate}}$$

$$= \frac{\text{Distance} \times \text{Bit rate}}{\text{Bits Transmitted} \times \text{Speed of Signal}}$$

Student Questions

- Why do you use the term "non-increasing" when describing the shape of the graph? Why not "constant"? Is there a specific reason?

Non-increasing = Decreasing + Constant

- Is efficiency the same as utilization?

They are different but equal if 100% of resources are used.

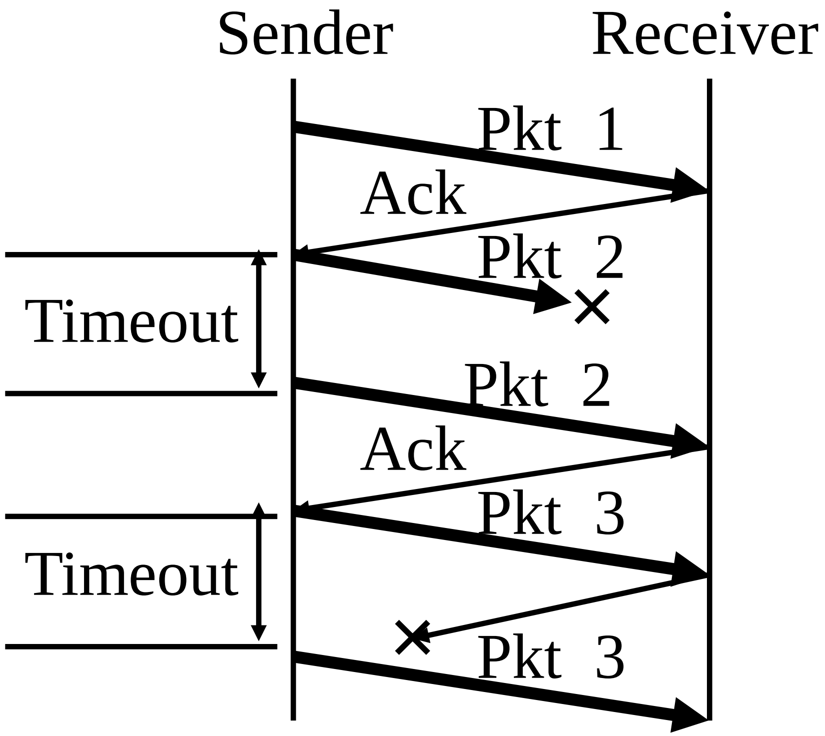
$$\text{Utilization} = \frac{\text{Output traffic}}{\textit{Capacity}}$$

$$\text{Efficiency} = \frac{\text{Output traffic}}{\text{Resources used}}$$

Error Control: Retransmissions

- ❑ Error Control = Error Recovery
- ❑ Retransmit lost packets ⇒ **A**utomatic **R**epeat re**Q**uest (ARQ)

Stop and Wait ARQ



Student Questions

- ❑ Is it possible that we thought a packet was lost after a timeout, and we resend it, then we received two Ack that this packet was received? If so, can we identify what packet is received from Ack?

This is quite common. Both packets are identical. We throw away one. But two acks are sent to ensure that the ack was not lost.

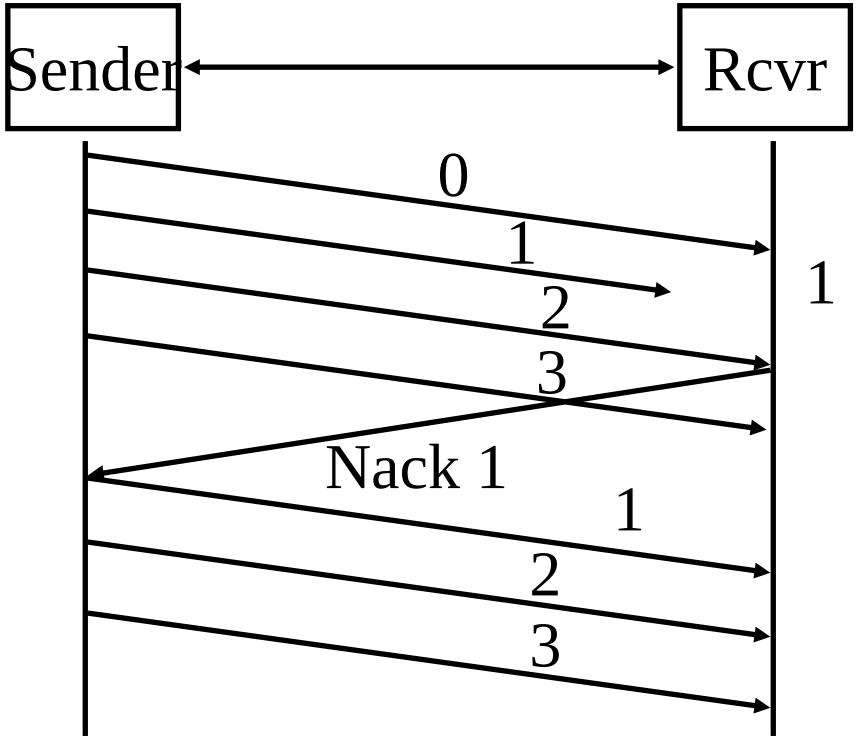
- ❑ If the acknowledgment is not received after a timeout, do we resend the packet immediately? *Yes. Some packets, but not all. Slow start is discussed later in this chapter.*
- ❑ What if the same packet is lost many times? Will it resend the packet every time? *After some number of retries, the connection may be disconnected.*
- ❑ Error Control is only for TCP, right?

UDP does not have error recovery.

- ❑ Does that mean the packet size is doubled with the introduction of error checking?

No. The checksum is very small (say, 4 bytes) compared to the data size (1024).

Go-Back-N ARQ



- The receiver does not cache out-of-order frames
- The sender has to ***go back*** and retransmit all frames after the lost frame.

Student Questions

- TCP has similarities with GBN/selective repeat but does not explicitly use these, correct?
- Is there a standard on how long a message is held on to after it is sent? Could the sender respond to a Nack 0 after Message 3 is sent?

Yes. The transmitter and receiver operate asynchronously. They react to events at their end.

- How does the receiver know which segment is lost?

It keeps track of the sequence numbers of received packets.

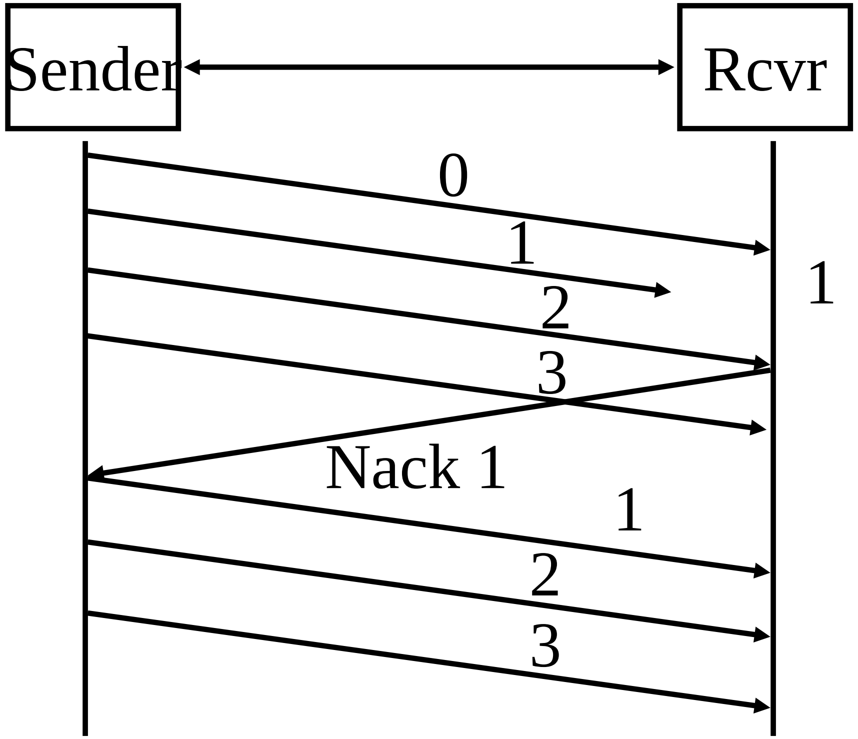
- Why don't we send something like "Nack <# of missed packet> <# of received packet>"?

This is called Selective Ack. It was added later.

- What is the advantage of Go-Back-N ARQ?

Receiver code is simplified

Go-Back-N ARQ



- ❑ The receiver does not cache out-of-order frames
- ❑ The sender has to ***go back*** and retransmit all frames after the lost frame.

Student Questions

- ❑ The book uses ACK 0 for receiving packet 0 instead of NACK 1. What is the standard?

We are discussing various possible forms of flow control. TCP designers selected one of these.

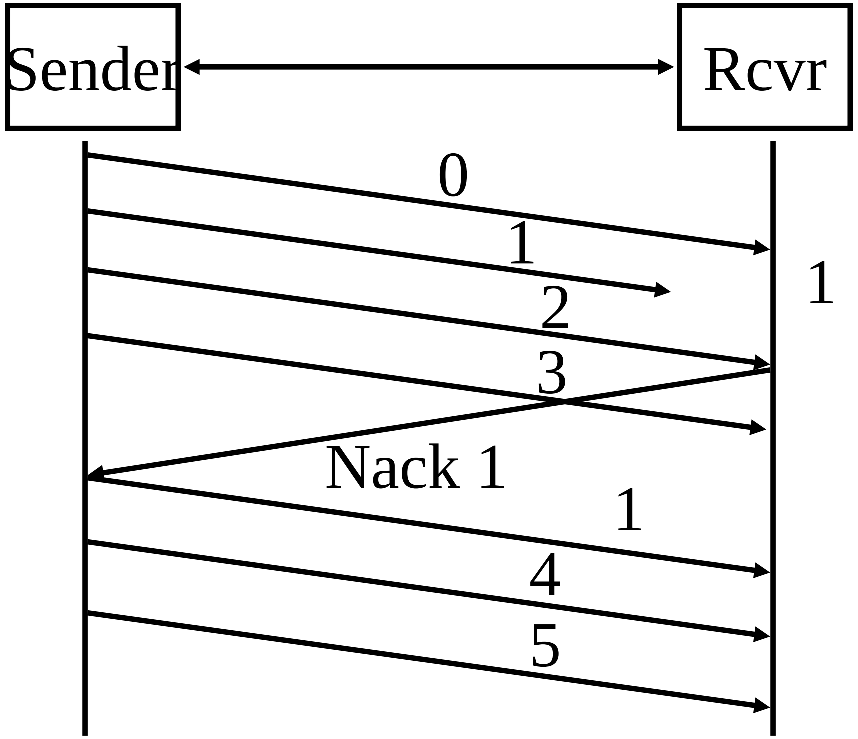
- ❑ There seem to be many ARQs. Is one preferred? I am assuming it is a very situation-dependent one.

Protocol designers select one based on the complexity (and, therefore, cost).

- ❖ Can we go over how GBN works again?

Sure.

Selective Repeat ARQ



- The receiver caches out-of-order frames
- The sender retransmits only the lost frame
- Also known as selective *reject* ARQ

Student Questions

- Can you nack and ack at the same time in Selective Repeat ARQ. How would the Sender know that frame 2 has been received if the receiver doesn't send an ack on 2, and sends an ack (for 3) on 3. Or can you ack more than one frame at once?

All acks are cumulative. If you ack n, than all packets up to n have been received. Nacks are cumulative too. When you nack n, then all packets up to n-1 have been received.

- How does the receiver know that it has missed frame 1? but don't know whether the next frame it has received is frame 2 or 3?

Every frame has a sequence number in it.

- Can selective-repeat protocol or go-back-n protocol with window size 1 used as alternating-bit protocol? *Yes.*
- What would happen if I lost both 1 and 2?

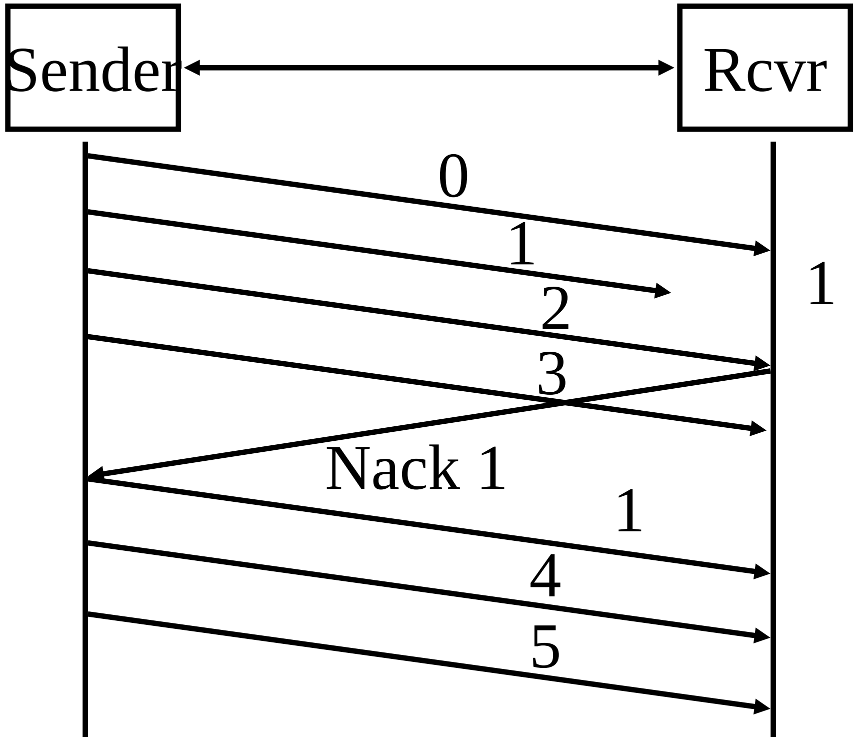
Nack 1 will be sent on receiving 3.

- Is there any incentive to use Go-Back N ARQ over selective repeat ARQ?

Go-back-N is simpler for the receiver.

- How can the receiver keep the packages in the correct order in this case? *Every byte is numbered.*

Selective Repeat ARQ



- The receiver caches out-of-order frames
- The sender retransmits only the lost frame
- Also known as selective *reject* ARQ

Student Questions

- What does cache out-of-order frame mean? How does that help the retransmission process?

Segments received after some missing segments are saved at the receiver. It reduces the number to be retransmitted.

- How do the sender and receiver know which ARQ they are using?

It is specified in the standard. If the standard specifies more than one, it is negotiated at connection setup.

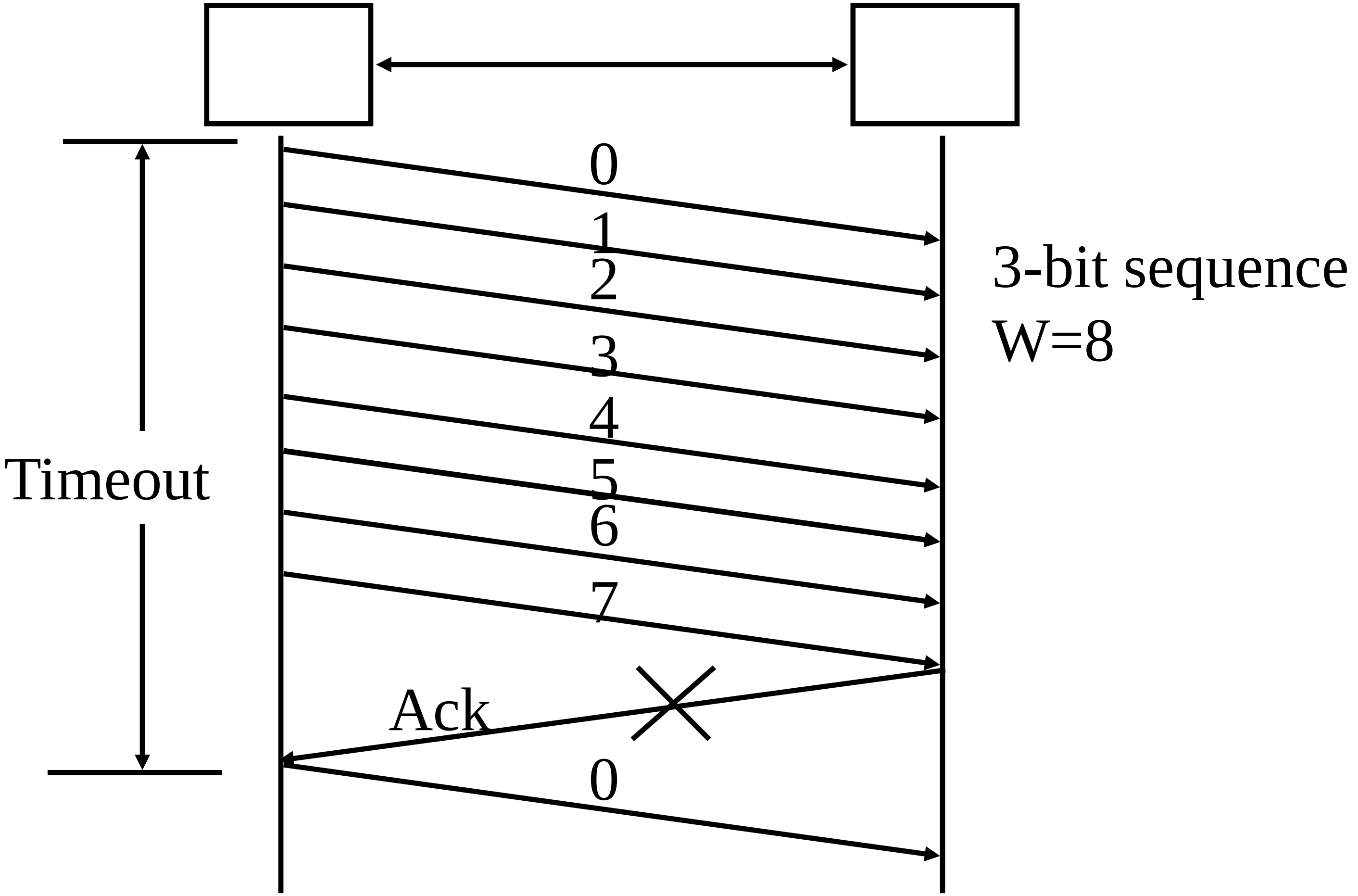
- If it gets two but not 0 and 1 or just Nack 0.

Nack 0.

- Is a NACK always just an ACK for the previously received packet? If a server NACKs a packet k, is the NACK always just an ACK segment with seqnum k - 1, or are there other ways to NACK packet k?

Yes. You could send a bit sequence of 0 and 1 to indicate which segments have been received and which are missing.

Selective Repeat: Window Size



Sequence number space ≥ 2 window size

Window size ≤ 2^{n-1} with n bit sequence numbers

Student Questions

- Is an ack sent for each received packet in a window?

No. Generally, the receiver waits until it receives either a sufficient number of packets or no packets are received for a certain interval before sending the ack.

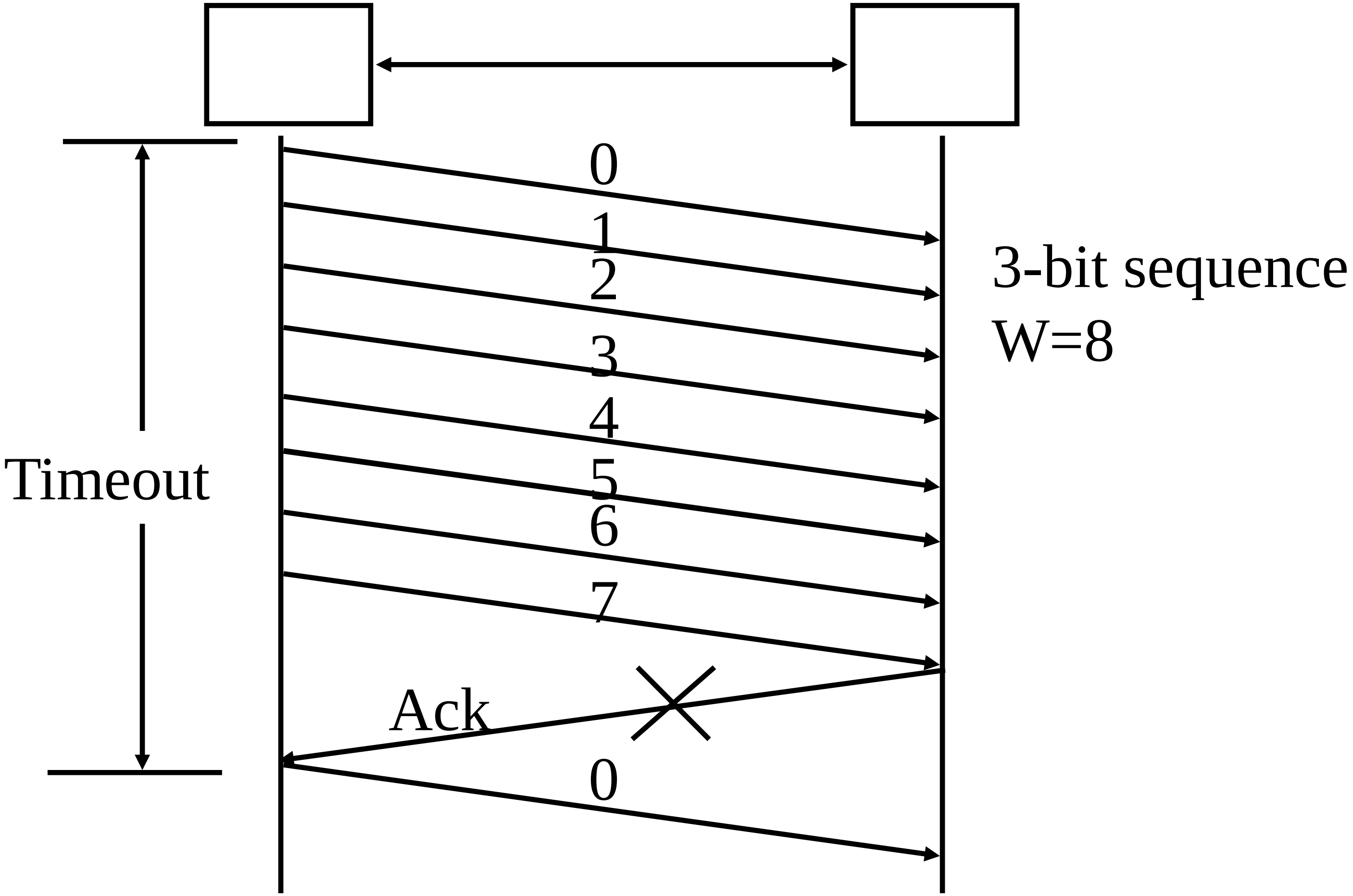
- How do we select the optimal window size if there are multiple possibilities?

Smallest Optimal

- In this case, does the missed packet get sent in the next window, or can it be sent in any of the upcoming windows? What would be the performance implication for each of these cases? *The missed packet is sent as soon as it is realized that the packet may have been lost. The window cannot advance beyond that permitted by the packet ack'ed.*
- What is sequence number? *Packet number*
- What is the unit of the window size?

It is packets, as shown here. But TCP uses bytes.

Selective Repeat: Window Size



Sequence number space ≥ 2 window size

Window size ≤ 2^{n-1} with n bit sequence numbers

Student Questions

- ❑ Why the sequence number space must be >= 2 * window size.

Otherwise, the packet numbers will repeat faster than the window.

- ❑ The frame sequence number is used by TCP protocol in flow control. Is there another sequence number for TCP protocol to notice the other end of the sequence of frames it should receive in order?

The flow control window specifies both the beginning and the end.

- ❑ What is an n-bit sequence number?

2-bit sequence numbers: 00, 01, 10, 11
3-bit sequence numbers:000, 001, 010, 100, 101, 110, 111.

- ❑ For the second packet 0, would that packet contain different information than the first packet 0? If an acknowledgement was received, or would the numbering continue, i.e., 8, 9, 10, ... are the following packets sent.

If an ack is received, that number can be reused after the numbers complete the cycle.

- ❑ Why must the window size be less than 2n-1?

You will not be able to distinguish some lost packets.

Performance: Maximum Utilization

- **Stop and Wait Flow Control**: $U = 1/(1+2\alpha)$
- **Window Flow Control**:

$$U = \begin{cases} 1 & W \geq 2\alpha+1 \\ W/(2\alpha+1) & W < 2\alpha+1 \end{cases}$$

- **Stop and Wait ARQ**: $U = (1-P)/(1+2\alpha)$
- **Go-back-N ARQ**:

P = Probability of Loss

$$U = \begin{cases} (1-P)/(1+2\alpha P) & W \geq 2\alpha+1 \\ W(1-P)/[(2\alpha+1)(1-P+WP)] & W < 2\alpha+1 \end{cases}$$

- **Selective Repeat ARQ**:

$$U = \begin{cases} (1-P) & W \geq 2\alpha+1 \\ W(1-P)/(2\alpha+1) & W < 2\alpha+1 \end{cases}$$

Student Questions

- How can we get the formula of go-back-N ARQ and selective repeat ARQ?

Good exercise. You can do it if you know probability theory.

- Do we need to remember these formulas for the exam?

Yes

- What is Stop and Wait ARQ? Isn't Stop and Wait a flow control strategy?

"Stop and Wait" is used for both error control and flow control.

- *How* are the U's for ARQs calculated?

ARQ includes the probability of error

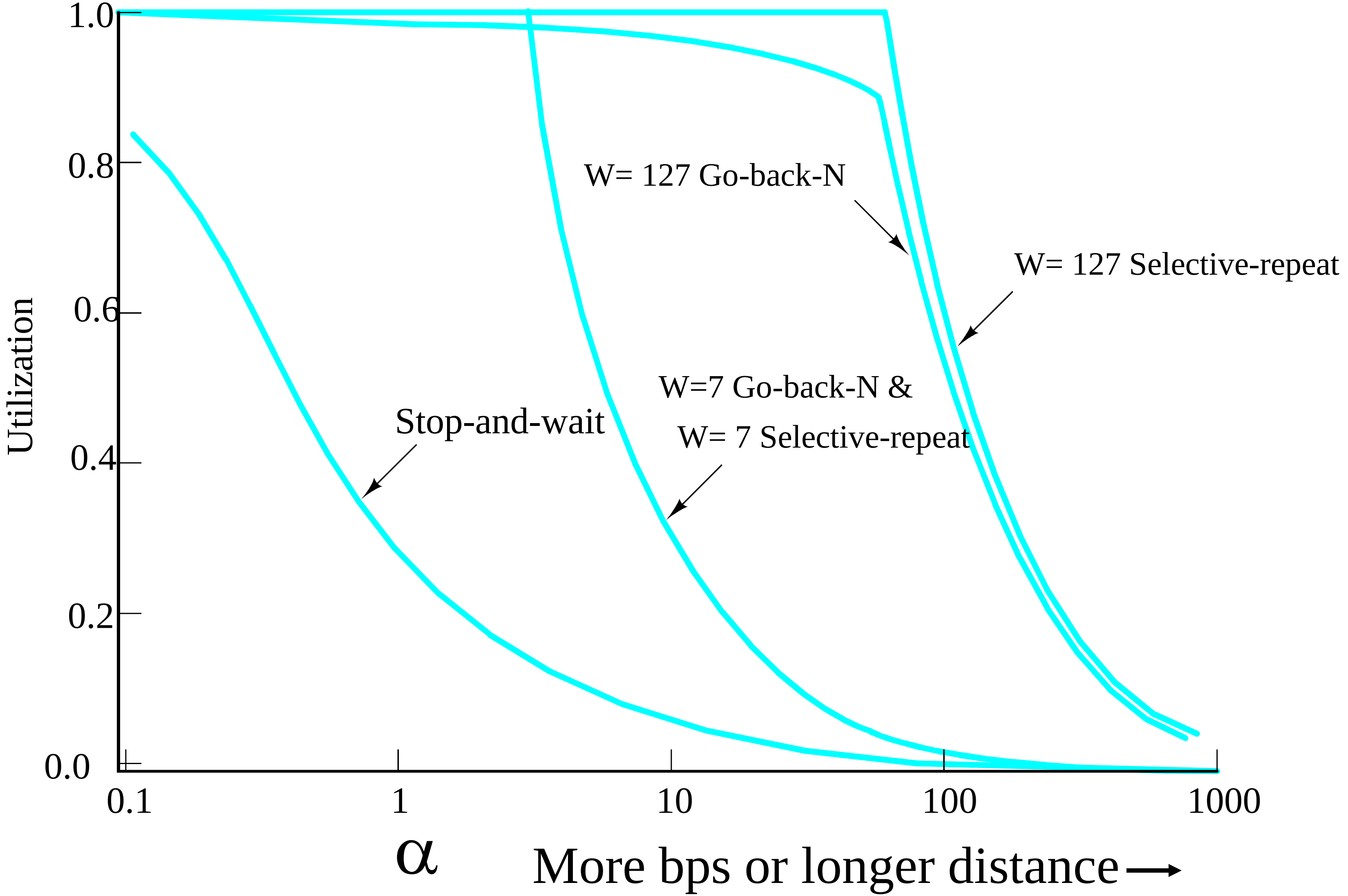
- Why is U different?

Different schemes allow different throughput and hence different utilization of the link.

- How do you find the total U with both flow control and ARQ?

Take the minimum.

Performance Comparison



Student Questions

- Why would someone use Stop and Wait ARQ when Selective repeat is better?

Selective repeat requires memory to store out-of-order packets.

Transport Layer Design: Summary

1. Multiplexing/demultiplexing by a combination of source and destination IP addresses and port numbers.
2. Window flow control is better for long-distance or high-speed networks
3. Longer distance or higher speed
   $\Rightarrow$ Larger α $\Rightarrow$ Larger window is better.
4. Stop and and wait flow control is ok for short-distance or low-speed networks
5. Selective repeat is better than stop and wait ARQ
   Only slightly better than go-back-N

Read Sections 3.4.2, 3.4.3, and 3.4.4. Do R11-13, P23, P24

Student Questions

Homework 3B: Flow Control

[8 points] Similar to problem 22 on page 292 of the textbook:

Consider the GBN protocol with a sender window size of 3 and a sequence number range of 1,024. Suppose that at time t, the next in-order packet that the receiver is expecting has a sequence number of k. Assume that the medium does not reorder messages. Answer the following questions:

A. What are the possible sets of sequence numbers inside the sender's window at time t? Justify your answer.

B. What are all possible values of the ACK field in all possible messages currently propagating back to the sender at time t? Justify your answer.

Window Flow Control:

C. How big a window (in the number of packets) is required for the channel utilization to be greater than 70% on a cross-country fiber link of 3000 km running at 40 Mbps using 1 kByte packets?

Efficiency Principle:

D. Ethernet V1 access protocol was designed to run at 10 Mbps over 2.5 Km using 1500-byte packets. This same protocol needs to be used at 100 Mbps at the same efficiency. What distance can it cover if the frame size is not changed?

Student Questions

- ❑ If the receiver expects k, does it mean it has already acked k - 1?

It means it has received everything up to and including k-1.

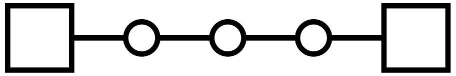
TCP

1. TCP Header Format, Options, Checksum
2. TCP Connection Management
3. Round Trip Time Estimation
4. Principles of Congestion Control
5. Slow Start Congestion Control

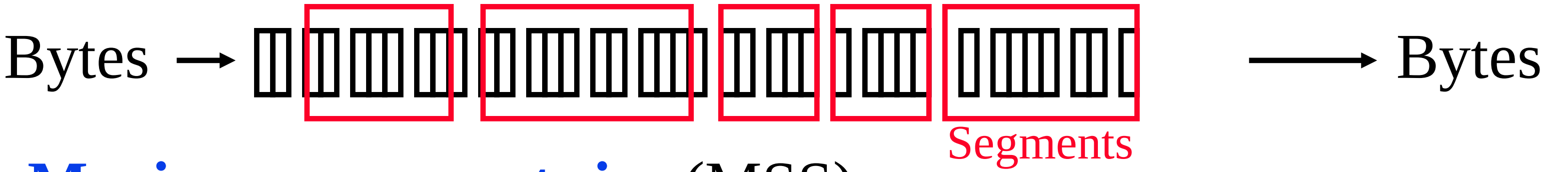
Student Questions

Key Features of TCP

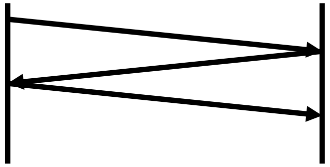
- ❑ **Point-to-Point**: One sender, one receiver



- ❑ **Byte Stream**: No message boundaries. TCP makes "segments"

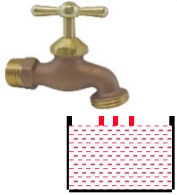


- ❑ **Maximum segment size** (MSS)
- ❑ **Connection Oriented**: Handshake to initialize states before data exchange



- ❑ **Full Duplex**: Bidirectional data flow in one connection

- ❑ **Reliable**: In-order byte delivery
- ❑ **Flow control**: To avoid receiver buffer overflow



- ❑ **Congestion control**: To avoid network router buffer overflow

Student Questions

❑ Are there any types of cyberattacks performed via overflowing a receive buffer?
Yes, buffer overflow attacks are quite common.

❑What is the maximum segment size for TCP?
Segment=TPDU. MSS is set by the application. The minimum MSS with IPv4 is 536B to allow sending one 512-byte block in one segment.

❑MSS limits the size of segments, so messages will be sliced into several segments. Does the same thing happen at the link layer and internet layer?
Yes. Every layer has its own maximum PDU size.

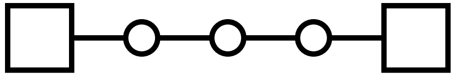
❑Regardless of the connection, what is the difference between the end-to-end service of UDP and the point-to-point of TCP?
Retransmission of lost PDUs.

❑Is the three-way handshake that makes TCP a connection-oriented protocol?
Yes.

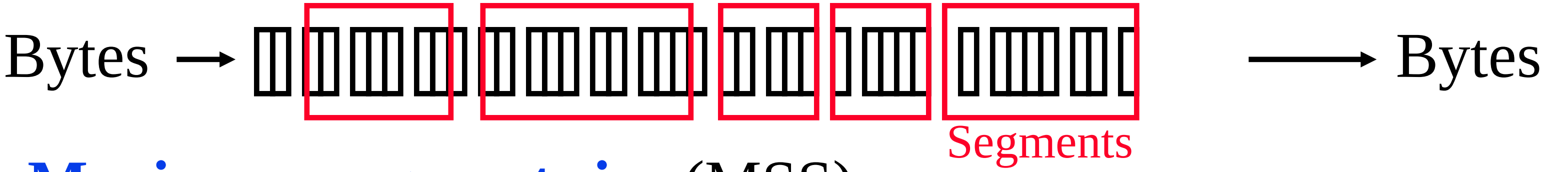
❑What determines the length of segments?
TCP decides to send a segment either when it has enough to send or after a set period.

Key Features of TCP

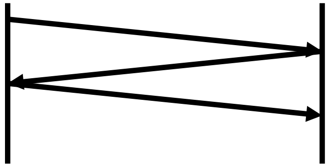
- **Point-to-Point**: One sender, one receiver



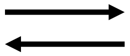
- **Byte Stream**: No message boundaries. TCP makes “segments”



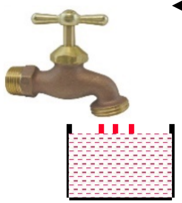
- **Maximum segment size** (MSS)
- **Connection Oriented**: Handshake to initialize states before data exchange



- **Full Duplex**: Bidirectional data flow in one connection



- **Reliable**: In-order byte delivery
- **Flow control**: To avoid receiver buffer overflow



- **Congestion control**: To avoid network router buffer overflow

Student Questions

- What are the limits of the TCP's guaranteed service? It is said that TCP allows for reliable data transfer over an unreliable network layer, but what happens when there's a large network outage? i.e., is TCP explicit about what conditions it requires to guarantee reliable transfer of data?

Yes, if there is a network outage, TCP will not deliver and so whatever is delivered is reliable.

- For review question R5 from the textbook, why does it say, "voice and video traffic is often sent over TCP rather than UDP"? I thought that videos were more often sent with UDP.

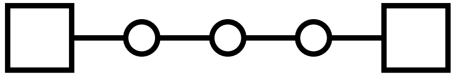
The only reason video traffic uses TCP is for the video parameters and control. The data itself is sent using UDP.

- How is the system able to serve both packets for UDP and bytes/segments for TCP?

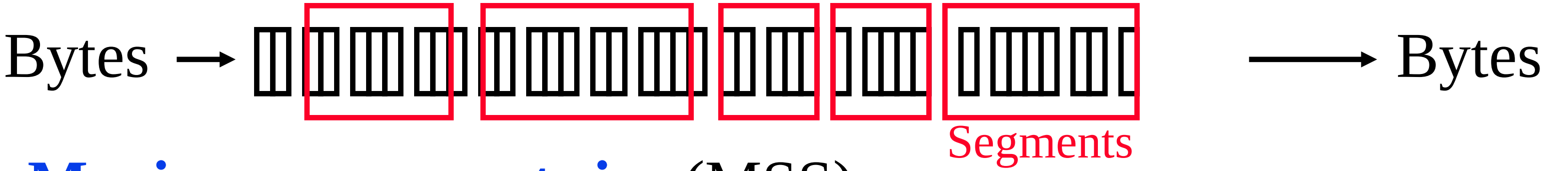
These are two different processes in the OS. IP delivers the packet to the right process by looking at the IP header.

Key Features of TCP

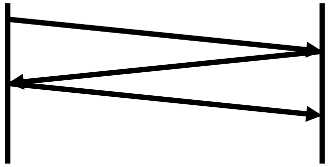
- **Point-to-Point**: One sender, one receiver



- **Byte Stream**: No message boundaries. TCP makes "segments"

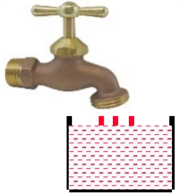


- **Maximum segment size** (MSS)
- **Connection Oriented**: Handshake to initialize states before data exchange



- **Full Duplex**: Bidirectional data flow in one connection

- **Reliable**: In-order byte delivery
- **Flow control**: To avoid receiver buffer overflow



- **Congestion control**: To avoid network router buffer overflow

Student Questions

- ❑ If congestion control only prevents router buffer overflow, why are they used on TCP since it runs only on the end system?

IP runs on routers. So IP tells TCP about the buffers, and TCP takes care of it.

- ❑ Does TCP always create a persistent connection?

Yes. Each connection has to be explicitly opened and closed.

- ❑ Are there any situations in which you wouldn't want TCP to establish a persistent connection?

Either connect, transact, and disconnect or use UDP.

- ❑ Who/what determines the segment size?

The receiver.

What is Congestion control?

Congestion=Load>capacity

- ❑ How many Congestion Control techniques are there?

See Slide 3.72

TCP

- Transmission Control Protocol
- Key Services:
  - **Send**: Please send when convenient
  - **Data stream push**: Destination TCP, please deliver it immediately to the receiving application.
    ⇒ Source TCP, please send it now.
    Set on the last packet of an application message.
  - **Urgent data signaling**: Destination TCP, please give this urgent data to the user out-of-band.
    Generally used for CTRL-C.

Student Questions

- If push is used, wouldn't that scramble the data in the stream? *No, everything is pushed.*
- Those key features seem to require the cooperation of intermediate hop, but since we didn't implement the transport layer in routers, how does the router know if this is urgent data or not?
  Routers do not distinguish urgent data or non-urgent data.
- Can we also simulate TCP transmission on our localhost?
  Yes. Two processes can send and receive via TCP—for example, a web server and browser on the same host.
- When using send, how is convenience determined?
  Convenience = Do I have enough, or has enough time passed?
- If CTRL-C is pressed, does it mean the remaining unsent data will never be sent?
  CTRL-C does not have any additional data associated with it. Other urgent tasks may have

TCP

- Transmission Control Protocol
- Key Services:
  - **Send**: Please send when convenient
  - **Data stream push**: Destination TCP, please deliver it immediately to the receiving application.
    ⇒ Source TCP, please send it now.
    Set on the last packet of an application message.
  - **Urgent data signaling**: Destination TCP, please give this urgent data to the user out-of-band.
    Generally used for CTRL-C.

Student Questions

- If CTRL-C is pressed, does it mean the remaining unsent data will never be sent?

CTRL-C does not have any additional data associated with it. Other urgent tasks may have

- Can you clarify the example for data stream push?

Push=Give it to the application.

- What is the difference between the destination TCP and Source TCP, and what if they are both requesting?

For each segment, there is a source and destination.

- How does urgent work to send data to the user out-of-band? Does broadcasting do it?

It is given even if earlier bytes have yet to arrive.

- How do we decide which method we should use?

Urgent is used only for abnormal conditions.

- What does out-of-band mean?

Out-of-line or out-of-order

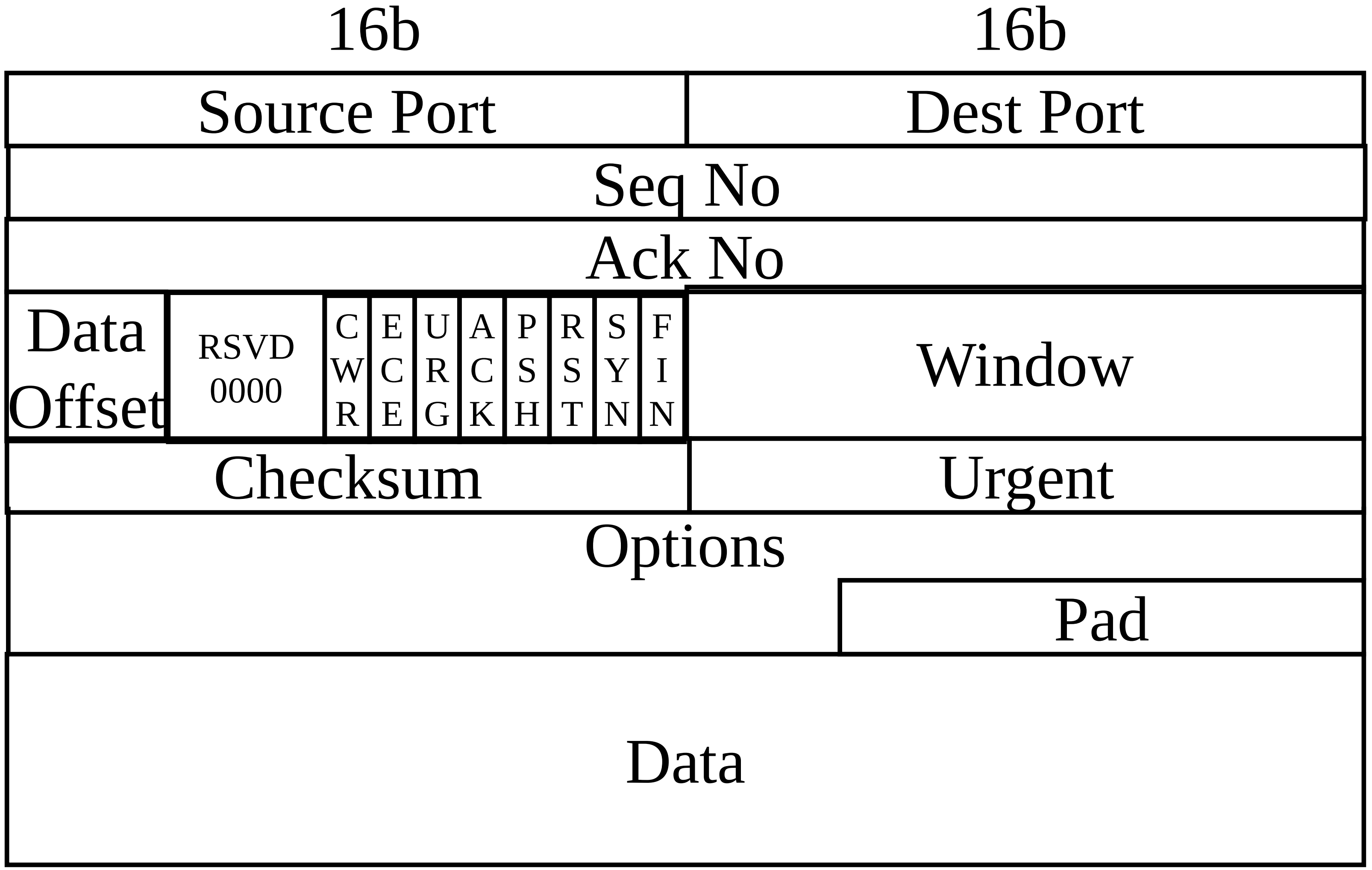
- Are there any other examples besides CTRL-C for urgent data?

Other types of aborts

- Are there other TCP services than these?

Yes. But you will need to see RFC 9293 for other services.

TCP Segment Format (Cont)



16b | 16b

Source Port | Dest Port

Seq No

Ack No

Data Offset | RSVD 0000 | CWR | ECE | URG | ACK | PSH | RST | SYN | FIN | Window

Checksum | Urgent

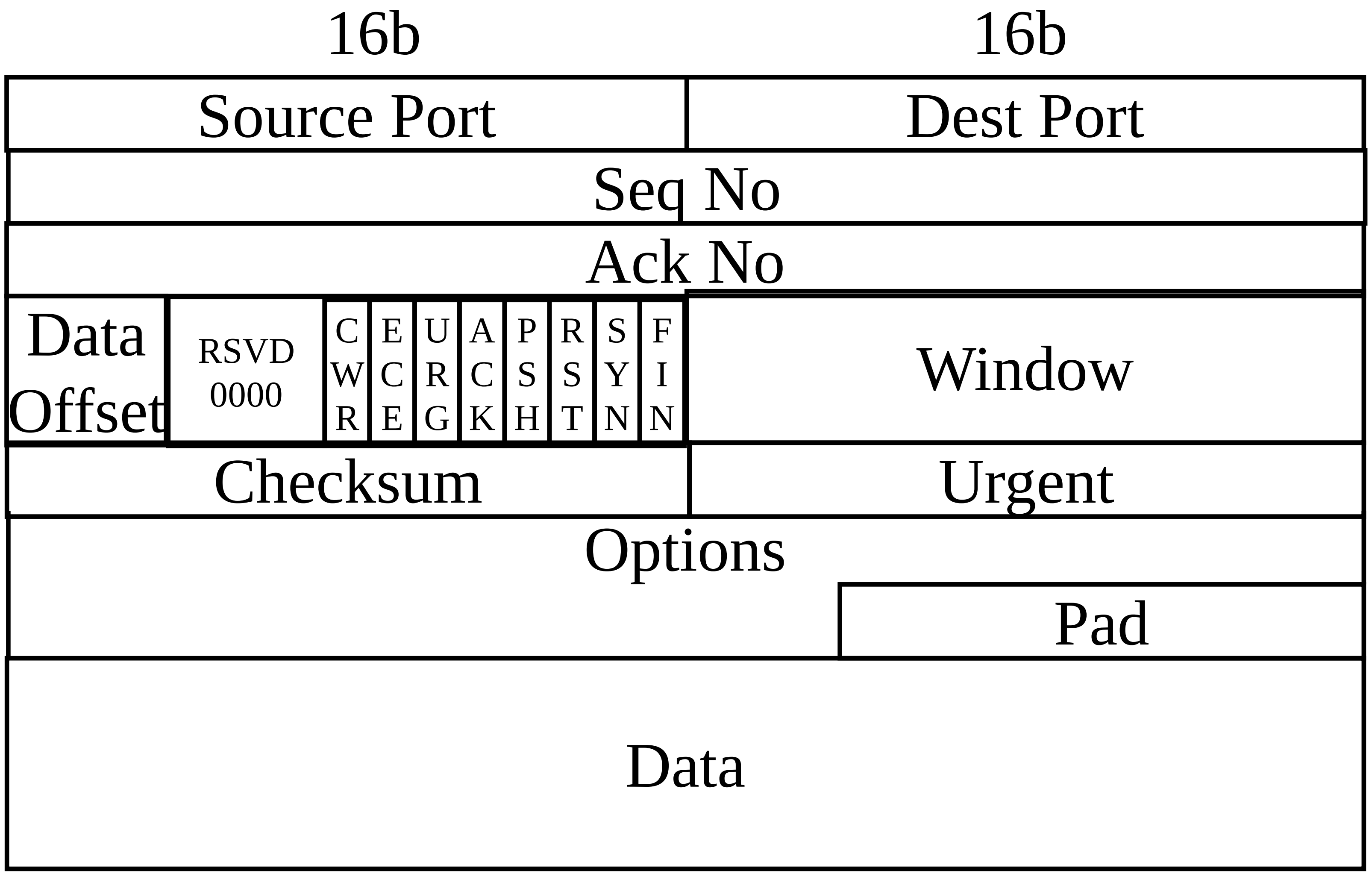
Options | Pad

Data

Student Questions

- ❑ What happens if I send more than 2^32 segments of data? Would the sequence and acknowledgment numbers overflow?
  The sequence number and acks "wrap around." This is not a problem if it happens after a large time.
- ❑ Can you explain this again? It's hard to understand what you're pointing at because the video is a screen recording.
  Sure.
- ❑ Will there ever be a time that the reserved bits get assigned to be used for something, or are they just never going to be used?
  Often. They use it for new features.
- ❑ Is the Pad used to fill up 32 bits or a multiple of 32 bits?
  The minimum number of bits required to make it a multiple of 32 bits.
- ❑ You mentioned that there are 15 words at most. Where does this number come from?
  The data offset is 4 bits. So max header=2^4-1

TCP Segment Format (Cont)



<table>
<tr><th colspan="10">16b</th><th>16b</th></tr>
<tr><td colspan="10">Source Port</td><td>Dest Port</td></tr>
<tr><td colspan="11">Seq No</td></tr>
<tr><td colspan="11">Ack No</td></tr>
<tr><td>Data Offset</td><td>RSVD 0000</td><td>CWR</td><td>ECE</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>Window</td></tr>
<tr><td colspan="10">Checksum</td><td>Urgent</td></tr>
<tr><td colspan="11">Options</td></tr>
<tr><td colspan="10"></td><td>Pad</td></tr>
<tr><td colspan="11">Data</td></tr>
</table>

Student Questions

- ❑ What is the header part in this format, everything before data?

Yes.

- ❑ Are the four reserved bits used for CWR and ECE bits discussed in the book?

Yes, Initially, there were six reserved bits. Two have been used for CWR and ECE.

- ❑ Where is urgent data?

In the data field.

- ❑ Where does the urgent data begin and end?

It ends at the byte pointed to by the urgent pointer.

- ❑ *Are there fillers after the urgent data?*

No, it is regular data or the end of data.

- ❑ *With TCP, is there any way to authenticate that the source port is where the packet comes from?*

Secure transports such as TSL do node authentication. We discuss this in Chapter 8.

- ❑ *What is RSVD?*

Reserved

- ❑ *How do you know when the syn bit is present to know when the numbering is the bit+1*

During connection setup.

- ❑ *Could you please explain the model again*

Sure.

TCP Header Fields

- **Source Port** (16 bits): Identifies source user process
- **Destination Port** (16 bits)
  21 = FTP, 23 = Telnet, 53 = DNS, 80 = HTTP, ...
- **Sequence Number** (32 bits): Sequence number of the first byte in the segment. If SYN is present, this is the initial sequence number (ISN), and the first data byte is ISN+1.
- **Ack number** (32 bits): Next byte expected
- **Data offset** (4 bits): Number of 32-bit words in the header
- **Reserved** (4 bits)

Student Questions

- Could you explain why, if syn is present, the first data byte is ISN+1?
  This is the first segment with data.
- Can we review what SYN refers to again?
  SYN=Synchronize Sequence Number
- Are the reserved bits being held for future use?
  Yes.
- Is the sender responsible for choosing the ISN?
  Yes.
- Could you explain again what exactly the data offset is used for? *Offset=Where does the data begin?*
- Why is it 16-bit?
  If you are asking about port numbers, they could not imagine any more than 64K applications.
- Could you explain how ISN works and why the sequence number will be set to ISN+1?
  ISN=Initial Sequence Number.
  Every byte is numbered. The first byte is numbered ISN+1.
- Can we get a sequence number of 0?
  Binary counting is from 0 to 2^n-1

TCP Header Fields

- ❑ **Source Port** (16 bits): Identifies source user process
- ❑ **Destination Port** (16 bits)
  21 = FTP, 23 = Telnet, 53 = DNS, 80 = HTTP, ...
- ❑ **Sequence Number** (32 bits): Sequence number of the first byte in the segment. If SYN is present, this is the initial sequence number (ISN), and the first data byte is ISN+1.
- ❑ **Ack number** (32 bits): Next byte expected
- ❑ **Data offset** (4 bits): Number of 32-bit words in the header
- ❑ **Reserved** (4 bits)

Student Questions

- ❑ What happens if we send segments to uncommon ports?

If a process has opened that port, the segment is given to that process. Otherwise, it is dropped.

- ❑ The Ack# for the TCP header is for the next byte expected; Does the Ack# for flow control mean the next package expected then?

Both are the next byte expected.

- ❑ Why TCP Acks the next byte expected but the flow control and ARQ discussed previously ACK the packet received?

ARQ discussed previously presented several possibilities. TCP designers chose one.

- ❑ Should we know what ports correspond to FTP, telnet, DNS, and http from memory?

Yes, Write it in your cheat sheet.

- ❑ Could you elaborate on how the "Reserved" bits work to introduce new features? Even if some new feature gets popular, there are only so many reserved bits, so it seems they would have filled up already.

Yes, that is possible. One reserved bit can be used to indicate the presence of a "set" of other fields.

TCP Header Fields

- **Source Port** (16 bits): Identifies source user process
- **Destination Port** (16 bits)
  21 = FTP, 23 = Telnet, 53 = DNS, 80 = HTTP, ...
- **Sequence Number** (32 bits): Sequence number of the first byte in the segment. If SYN is present, this is the initial sequence number (ISN), and the first data byte is ISN+1.
- **Ack number** (32 bits): Next byte expected
- **Data offset** (4 bits): Number of 32-bit words in the header
- **Reserved** (4 bits)

Student Questions

- What happens if we send segments to uncommon ports?

If a process has opened that port, the segment is given to that process. Otherwise, it is dropped.

- The Ack# for the TCP header is for the next byte expected; Does the Ack# for flow control mean the next package expected then?

Both are the next byte expected.

- Is the ACK No in a TCP segment only used if there is a two-way sending of data in a connection?

No. It is used even if there is a one-way flow of data.

TCP Header (Cont)

- **Control** (8 bits): Congestion Window Reset
  Explicit Congestion Experienced
  Urgent pointer field significant,
  Ack field significant,
  Push function,
  Reset the connection,
  Synchronize the sequence numbers,
  No more data from sender

CWR	ECE	URG	ACK	PSH	RST	SYN	FIN

- **Window** (16 bits):
  Will accept [Ack] to [Ack]+[window]-1

Student Questions

- What does it mean to synchronize the sequence numbers?

So the receiver knows what byte numbers the sender is going to send.

- When is the Ack field significant?

Ack=0 ⇒ Ignore the Acknowledgement number field.

- TCP accepts segments from 'ACK to [ACK+WINDOW-1]. Why is there a -1?

The count is from x to x+w-1. The count includes the first and the last number. There are w bytes, e.g., 2,3,4 is three bytes. 4=2+3-1

- Are the control bits mandatory and always have at least one bit set in a TCP packet?

All 8 bits are mandatory. All bits are independently set.

TCP Header (Cont)

- **Checksum** (16 bits): covers the segment plus a pseudo-header. Includes the following fields from the IP header: source and dest adr, protocol, and segment length. Protects from IP misdelivery.
- **Urgent pointer** (16 bits): Points to the byte following urgent data. It Lets the receiver knows how much data it should deliver right away out-of-band.
- **Options** (variable):
  Max segment size (does not include TCP header, default 536 bytes), Window scale factor, Selective Ack permitted, Timestamp, No-Op, End-of-options.

Student Questions

❑ Should we assume the max segment size is 536 bytes if it comes up in the future?

Yes, iff not specified. Iff =If and only if

❑ Why is 536 bytes default MSS? Can it be more?

MSS can be any number. To allow 512-byte blocks, they selected 536, which includes the application header. 9K and 64K are commonly used since file sizes are now large.

❑ Could you clarify why the TCP header has so many fields? Are all of them needed?

Yes. More have been added.

❑ This TCP checksum seems to consider more fields than the checksum that takes only 16-bit words before. Is the TCP checksum another checksum than before, or is it just the full description now?

The TCP checksum is the same size as the checksum discussed before. It is the same checksum. There is no difference.

TCP Options

Kind	Length	Meaning
0	1	End of Valid options in header
1	1	No-op
2	4	Maximum Segment Size
3	3	Window Scale Factor
8	10	Timestamp

- **End of Options**: Stop looking for further option
- **No-op**: Ignore this byte. Used to align the next option on a 4-byte word boundary
- **Max Segment Size (MSS):** Does not include TCP header

Student Questions

- What is the pseudo-header?

A virtual header that includes some IP information. We will see it in Layer 3.

- Could you re-explain the chart? From my understanding, the kind is the type in the TLV. So if the type value is 3, then the length will be 3, and the meaning of the value is the window scale factor.

Correct.

- Can we interpret the table as the first column would be the type of options, each with a different number; the second column would be the length of the option, which will be in bytes?

Yes

- What is the "pad" block that is included in the "options" block

To align at a 32-bit boundary.

- What's the difference between No-op and regular padding?

Padding is usually done at the end. No-Op is used in the middle.

TCP Checksum

- ❑ The checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the TCP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.
- ❑ The checksum field is filled with zeros initially.
- ❑ TCP length (in octet) is not transmitted but used in calculations.
- ❑ Efficient implementation in RFC1071.

Source Adr	Dest. Adr	Zeros	Protocol	TCP Length
32b	32b	8b	8b	16b

TCP Header	TCP data

Student Questions

❑ Can you explain why the checksum is initially filled with zeros again?

Suppose A…B are message words. C is the checksum. A+…+B+C=0

C=-(A+..+B)

At the source:

A+..+B+0 = -C

At the destination: A+..+B+C=0

❑ Is there any difference between UDP Checksum and TCP Checksum? *No*

❑ In computing checksum, how do you add the source address to the protocol as they are of different bit lengths?

All IPv4 addresses are 32-bit long.

❑ The checksum for UDP was optional, but it's required for TCP, right? *Yes.*

❑ Is "zero octets" meant to be plural? Or is only one zero octet ever added for padding?

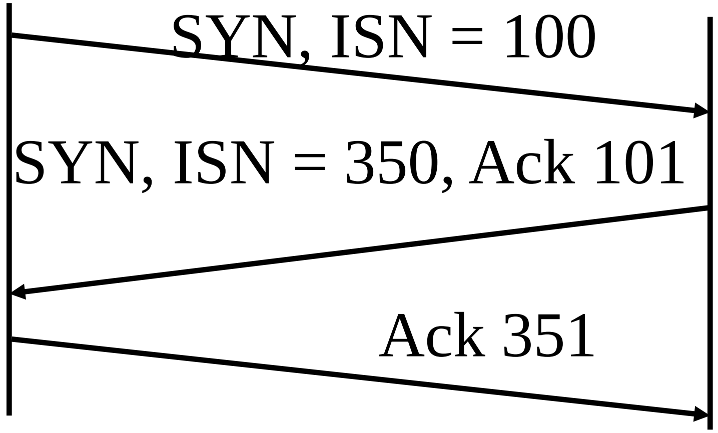
The padded octets have zeros in them.

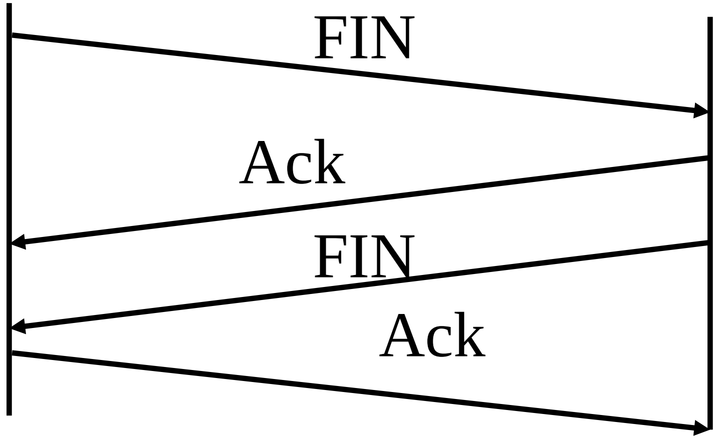
❑If checksum is found to be wrong will the entire connection be stopped or just that packet.

Just that packet will be dropped.

TCP Connection Management

- ❑ Connection Establishment
  - ➢ Three-way handshake
  - ➢ SYN flag set
    ⇒ Request for connection
- ❑ Connection Termination
  - ➢ Close with a FIN flag set
  - ➢ Abort





Student Questions

❑ What settings must be configured to prevent against a SYN Flood Attack?
Start closing half-open as the resources decrease. Connections

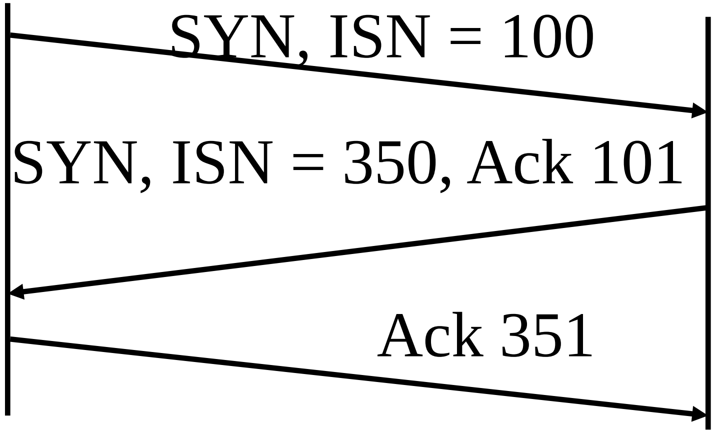
❑ When a client sends a SYN segment, do they also have to specify the port number to which they want to connect to? *Yes.*

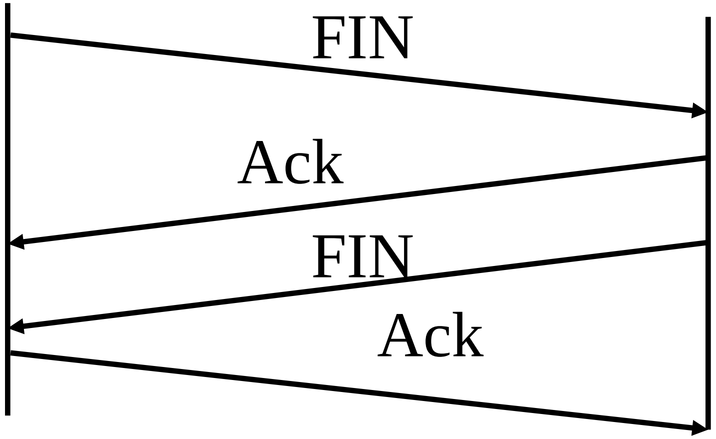
❑ If the first byte of a side (call this side A) starts at 101, why does the other side (call this side B) respond "ack 101," which means everything up to 101 was received? Does this mean side A will send the bytes from 101 only after receiving "ark 101" from B?
"Ack 101" for TCP designers meant that I was expecting 101. I have received everything up to 100.

❑ Could you please go over the connection establishment and the importance of the SYN flag set?
SYN ⇒ The sequence number field contains the initial sequence number.
What does the slide mean by a three-way handshake?
Three messages in the handshake

TCP Connection Management

- ❑ Connection Establishment
  - ➢ Three-way handshake
  - ➢ SYN flag set
    ⇒ Request for connection
- ❑ Connection Termination
  - ➢ Close with a FIN flag set
  - ➢ Abort





Student Questions

❑ What if the packet carrying FIN is lost? How will the client detect this situation?

The source of FIN will timeout and retransmit.

❑ Is the ack necessary for connection establishment since they've already exchanged seq numbers? What benefit does the receiver get after receiving the ack?

Three-way is necessary. It tells the right side that I know your ISN.

❑ Why do we need four steps (instead of 3) when terminating a connection?

Each FIN has to be acked.

❑ Does connection termination use a two-way handshake?

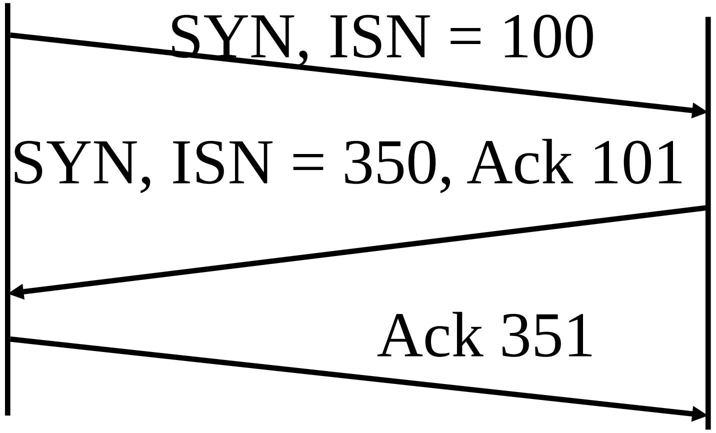
Four-way. But they are not sequential. There could be other segments between the two FINs

❖ Why are there two separate client and server ISNs?

Initial sequence numbers at each node start at a number higher than that used at that node.

TCP Connection Management

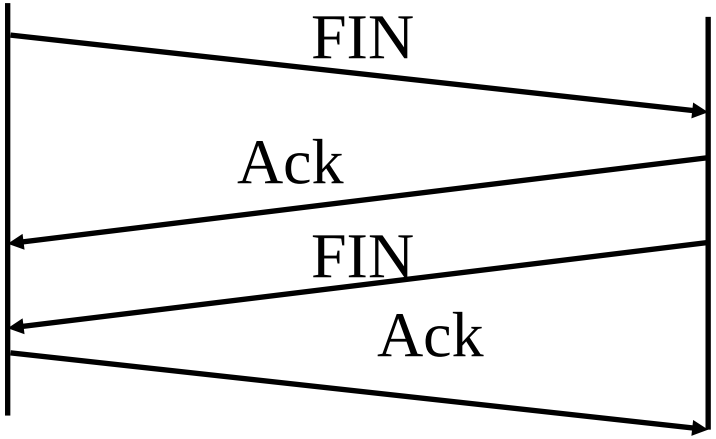
- Connection Establishment
  - Three-way handshake
  - SYN flag set
    ⇒ Request for connection
- Connection Termination
  - Close with a FIN flag set
  - Abort



SYN, ISN = 100

SYN, ISN = 350, Ack 101

Ack 351



FIN

Ack

FIN

Ack

Student Questions

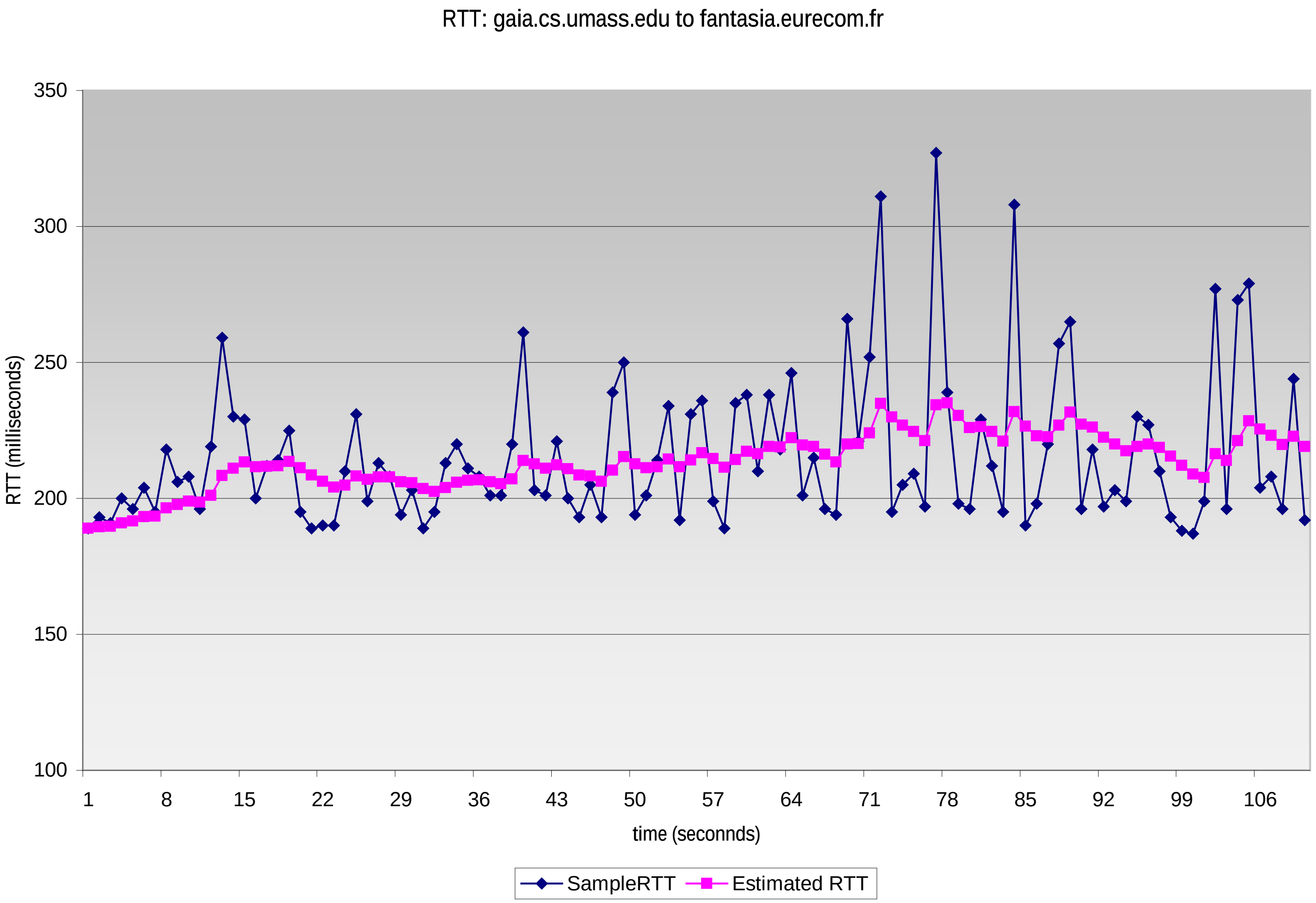
- If you were trying to perform an SYN Flood attack, could you send connection requests faster than half-open connections can be closed and thus still prevent new connections?

Yes. That is a SYN flood attack.

- Is the ISN always set to 100 during a connection negotiation?

No. It is an increasing number. Higher than any used in any recent previous connection.

Example RTT estimation:



Student Questions

- ❑ What is the relationship between RTT and propagation and transmission delay time?

RTT=2×Propagation+ Transmission + Queueing

- ❑ In the video, you mention that you "don't have to send the data." Was this referring to the first or second Ack? Also, why is this the case?

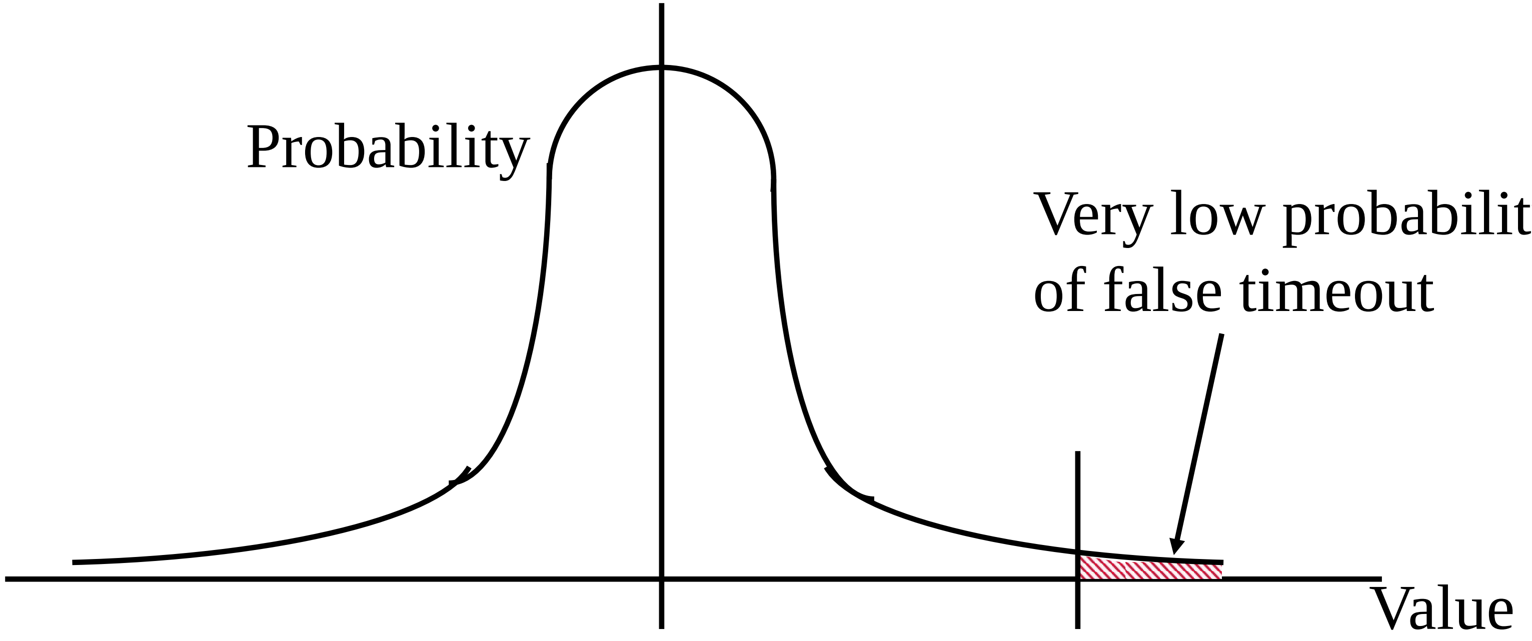
Ack packets may or may not have data going in the reverse direction.

- ❑ Why does sampleRTT deviate from EstimatedRTT at many points?

The sample is random. The estimate is smooth.

Round Trip Time Estimation

- Measured round trip time (SampleRTT) is very random.
- EstimatedRTT=(1- α)EstimatedRTT+α SampleRTT
- DevRTT = (1-β)DevRTT+ β |SampleRTT-EstmatedRTT|
- TimeoutInterval=EstimatedRTT+**4** DevRTT



Student Questions

❑ The book says the recommended value of alpha is 1/8. Where does this come from/ who decides this?

Determined by experiments at the time.

❑ To clarify, this chart shows the probability of incorrectly measuring RTT/timeout based on the value of a and b?

Actual RTT is a random number. The red area shows the probability of RTT being more than Timeout ⇒ Packet was not lost but assumed lost.

❑ Where do we get alpha and beta from?

Alpha and beta are parameters set by TCP users. Default values are normally used.

❑ Is RTT likely to follow a normal distribution?

Assumption

❑ Will there be any formula sheet on our exam that includes these?

Please see Video 0 on the course website

❑ Why is sample RTT random?

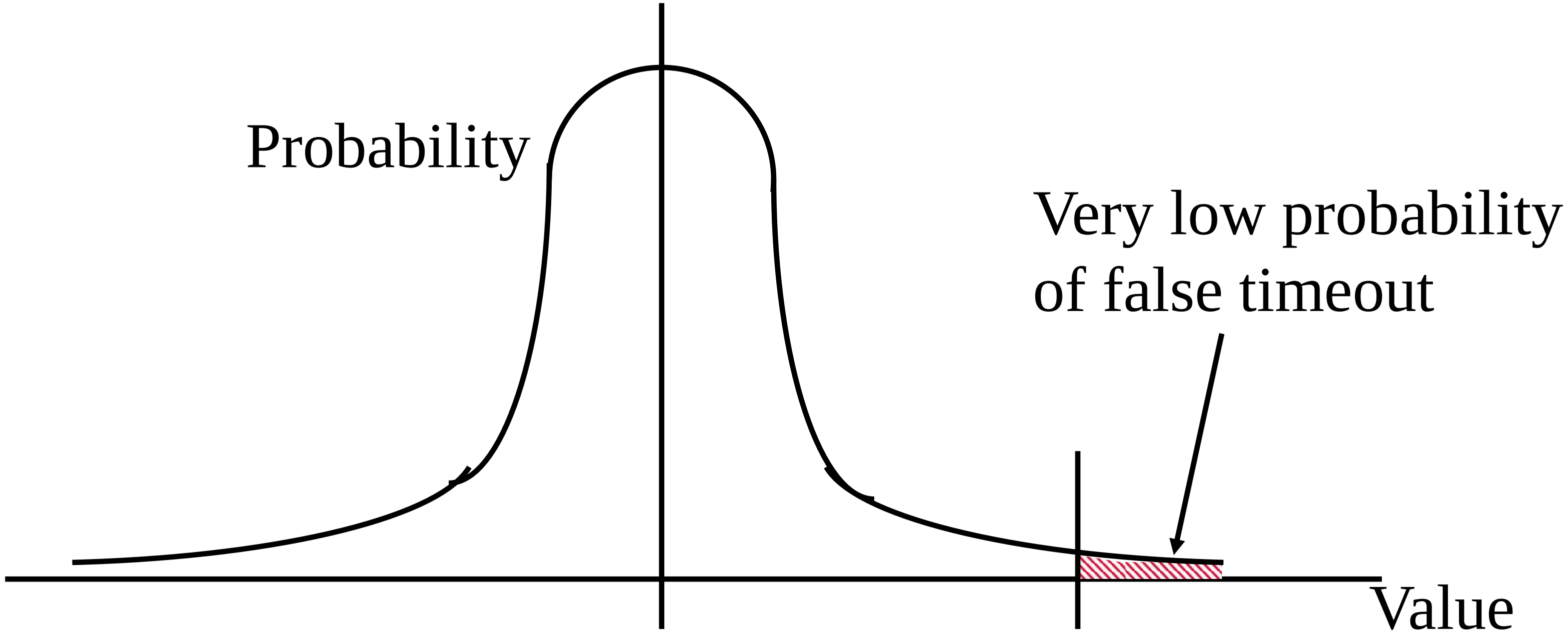
Because network traffic is random.

❑ Does RTT follow Gaussian distribution?

Not necessarily. It is assumed.

Round Trip Time Estimation

- Measured round trip time (SampleRTT) is very random.
- $\text{EstimatedRTT}=(1-\alpha)\text{EstimatedRTT}+\alpha\ \text{SampleRTT}$
- $\text{DevRTT} = (1-\beta)\text{DevRTT}+\beta\ |\text{SampleRTT}-\text{EstmatedRTT}|$
- $\text{TimeoutInterval}=\text{EstimatedRTT}+\mathbf{4}\ \text{DevRTT}$



Student Questions

- ❑ In the round trip time estimation, what does "α" and "β" stand for?

These are parameters selected by the network implementors or network managers. It must be between 0 and 1.

≈1 ⇒ Estimate=Sample. More random, but current

≈ 0 ⇒ More stable but less current

- ❑ Is there a way to mathematically find the optimal alpha and beta values?

Yes, with assumptions and desired probabilities.

- ❑ What do alpha and beta stand for?

They are "exponential weighting parameters."

- ❑ Could you explain this RTT probability curve?

Gaussian/Normal probability density function

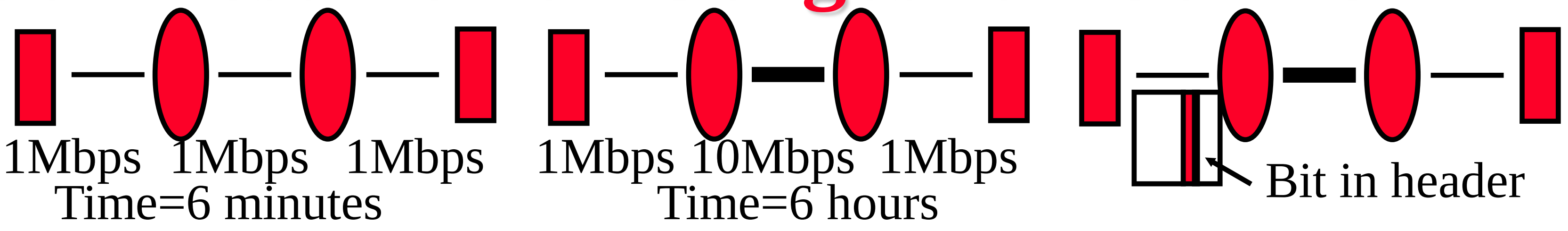
Is there a typo in the DevRTT equation? Should (1- β) be multiplied by EstimatedRTT instead of DevRTT?

No, there is no error. DevRTT is the estimated standard deviation, which depends on the difference.

How does the TCP timestamp option facilitate RTT measurement and avoid old duplicate segments?

"Timestamp" is basically a sequential number. While sending, the sender notes the number and clock time sent in a table. The receiver returns the number in the next outgoing packet. The sender then calculates the time difference.

Our Research on Congestion Control



1Mbps 1Mbps 1Mbps
Time=6 minutes

1Mbps 10Mbps 1Mbps
Time=6 hours

Bit in header

- Early 1980s, Digital Equipment Corporation (DEC) introduced Ethernet products
- Noticed that throughput goes down with a higher-speed link in the middle (because there are no congestion mechanisms in TCP)
- Results:
  1. Timeout ⇒ Congestion
     ⇒ Reduce the TCP window to one on a timeout [Jain 1986]
  2. Routers should set a bit when congested (DECbit).
     [Jain, Ramakrishnan, Chiu 1988]
  3. Introduced the term “Congestion Avoidance.”
  4. Additive increase and multiplicative decrease (AIMD principle)
     [Chiu and Jain 1989]
- There were presented to IETF in 1986.
  ⇒ Slow-start based on Timeout and AIMD [Van Jacobson 1988]

Student Questions

- Is the total speed always dependent on the link with the lowest speed? *Yes.*
- Could you talk about congestion again?
  Sure.
- Does the congestion occurs on the node that receives from the high-speed link and is sending it onto the low-speed link?
  Yes.
- Is there a significant difference in efficiency between setting a congestion bit and assuming there is congestion when there is a timeout?
  Yes. Timeout happens when a segment is lost. Setting the bit prevents that.
- Can you go over this slide again in class?
  Sure.
- Why does throughput go down with a higher-speed link in the middle? Shouldn't it be able to transmit more data?
  That’s precisely the point to note.

Slow Start Congestion Control

- Window = Flow control avoids receiver overrun
- Need congestion control to avoid network overrun
- The sender maintains two windows:
  Credits from the receiver
  Congestion window from the network
  The congestion window is always less than the receiver window
- Starts with a congestion window (CWND) of 1 max segment size (MSS)
  ⇒ Do not disturb existing connections too much.
- Increase CWND by 1 MSS every time an ack is received
- Assume CWND is in bytes

Student Questions

- You said multiple times that the packet size increases additively in "slow start," but your description only sounds exponential. Could you elaborate more on which ways the system is exponential and which ways it is additive?

The window is exponential in Time. The increase is additive per ack T=0: W=1
In 1st RTT: 1 Ack ⇒ W=2
In 2nd RTT: 2 Acks ⇒ W=4 In 3rd RTT: 4 Acks ⇒ W=8

- Is window size controlled by the number of ACKs received?

No. It is controlled by the window size permitted by the receiver.

- How do you find MSS?

Set by the node manager.

- How does the congestion window consider other hosts that may be trying to connect with the same server and use buffer space?

There is another formula to divide the resources among different clients.

Slow Start Congestion Control

- Window = Flow control avoids receiver overrun
- Need congestion control to avoid network overrun
- The sender maintains two windows:
  Credits from the receiver
  Congestion window from the network
  The congestion window is always less than the receiver window
- Starts with a congestion window (CWND) of 1 max segment size (MSS)
  ⇒ Do not disturb existing connections too much.
- Increase CWND by 1 MSS every time an ack is received
- Assume CWND is in bytes

Student Questions

- How does TCP's Slow Start manage CWND to balance resource allocation among multiple hosts?

There is no rule. You can give as much to any connection as you like and even overbook if you like.

- What's the difference between a congestion window, sender window, and receiver window, and are they managed separately?

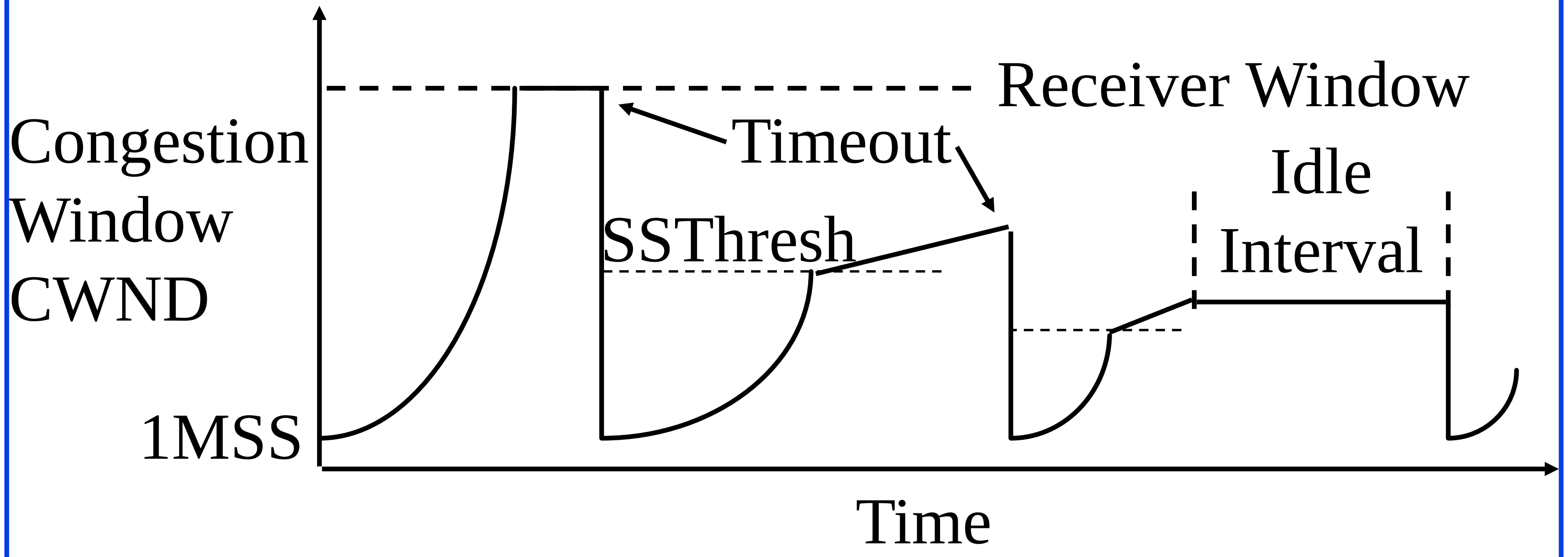
Congestion Window: A number maintained at the sender.

Sender Window: The current window used by the sender.

Receiver Window: Window allowed by the receiver

Slow Start (Cont)

- If segments lost, remember slow start threshold (SSThresh) to CWND/2
  Set CWND to 1 MSS
  Increment by 1MSS per ack until SSThresh
  Increment by 1 MSS*MSS/CWND per ack afterwards



Student Questions

- ❑ How is the Idle interval different?

No segments expected or sent

- ❑ This slides says that the window size increase by 1 MSS each **time**. The next slide says the window size increase by its original window size, which leads to exponential growth. How?

Please distinguish between "each ack" and "each roundtrip." There are W acks in each roundtrip.

- ❖ Can we please go over slow start one more time.

Sure.

Slow Start (Cont)

- At the beginning, SSThresh = Arbitrarily high value
- After a long idle period (exceeding one round-trip time), reset the congestion window to one.
- If CWND is W MSS, W acks are received in one round trip.
- Below SSThresh, CWND is increased by 1MSS on every ack
  ⇒ CWND increases to 2W MSS in one round trip
  ⇒ CWND increases exponentially with time
  Exponential growth phase is also known as "***Slow start***" phase
- Above SSThresh, CWND is increased by MSS/CWND on every ack
  ⇒ CWND increases by 1 MSS in one round trip.
  ⇒ CWND increases linearly with time.
  The linear growth phase is known as "***congestion avoidance***" phase

Student Questions

- What is meant by setting SSThresh initially to an "arbitrarily high value."

I checked many different sources, and here is the summary:

RFC 2001 - Dated Jan 1997 - says initially SSThresh should be set to 64kB.

This RFC is the first RFC describing Slow Start. However, it has been obsoleted by subsequent changes.

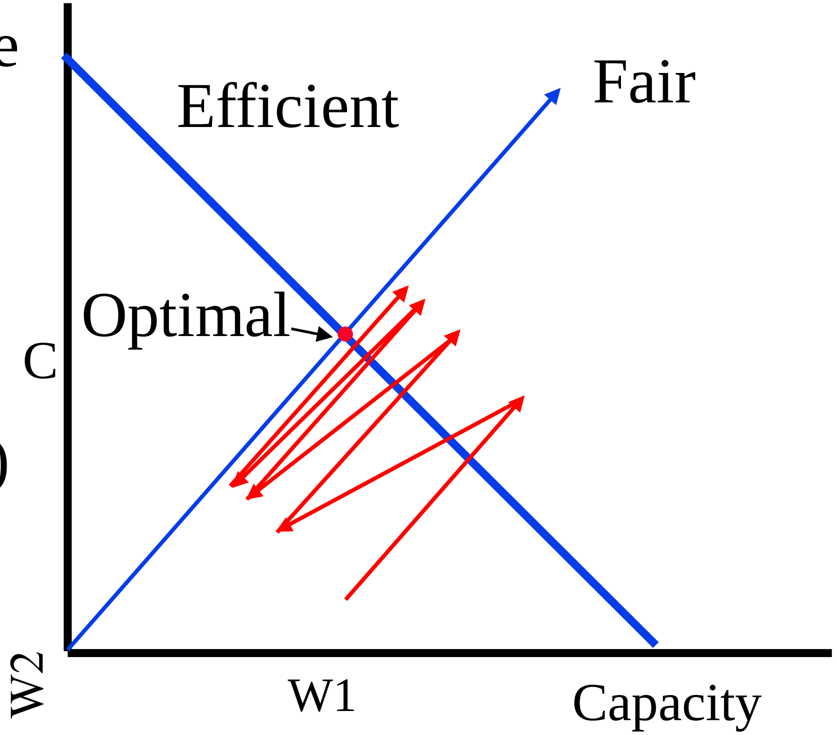
RFC 5681 - Dated Sept 2009 - says initially SSThresh should be set to arbitrarily high.

What changed in those 12 years? The networks became faster, and 64kB, which was very high in 1997 became small, and so no number was removed.

However, all documents state that the CWND can never exceed the Receiver Window. So slow start continues up to the receiver window if there is no loss. Setting to arbitrarily high value covers the case when the receiver window is changed over time. In this case, the slow start will continue until the point when CWND is equal to the receiver window.

AIMD Principle

- ❑ Additive Increase, Multiplicative Decrease
- ❑ W1+W2 = Capacity
  ⇒ Efficiency,
  W1=W2 ⇒ Fairness
- ❑ (W1,W2) to (W1+ΔW,W2+ΔW)
  ⇒ Linear increase (45° line)
- ❑ (W1,W2) to (kW1,kW2)
  ⇒ Multiplicative decrease
  (line through origin)



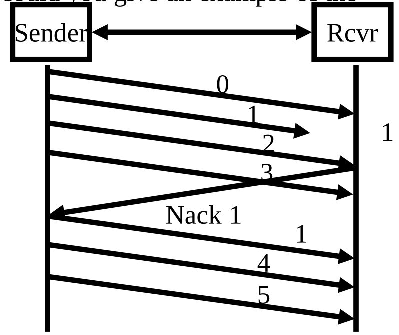
Ref: D. Chiu and Raj Jain, "**Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks**," Journal of Computer Networks and ISDN, Vol. 17, No. 1, June 1989, pp. 1-14, http://www.cse.wustl.edu/~jain/papers/cong_av.htm

Student Questions

- ❑ Could you explain the graph again, as the video doesn't show where you're pointing on the board; will it always reach an equilibrium where efficiency and fairness intersect?

The optimal is the intersection of the fair and efficient line shown by the red dot.
Yes, AIMD will always take you to the optimal point.

- ❑ For duplicate acks, could you give an example of the lost segment?



- ❑ What is different between different generations of TCP?

Too many to list. New features are added every year.

- ❑ What are the red lines?

The lines show the path of the two window variables W1 and W2

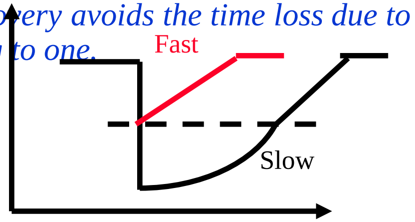
Fast Retransmit

- ❑ Optional – implemented in TCP Reno
  (Earlier version was TCP Tahoe)
- ❑ Duplicate Ack indicates a lost/out-of-order segment
- ❑ On receiving 3 duplicate acks (4th ack for the same segment):
  - ➢ Enter Fast Recovery mode
    - ❑ Retransmit missing segment
    - ❑ Set SSThresh=CWND/2
    - ❑ Set CWND=SSThresh+3 MSS **(Note: CWND is inflated)**
    - ❑ Every subsequent duplicate ack: CWND=CWND+1MSS
  - ➢ When a new ack (not a duplicate ack) is received
    - ❑ Exit fast recovery
    - ❑ Set CWND=SSTHRESH **(Note: CWND is deflated back**)

Student Questions

❑ What exactly is fast recovery doing? is it simply a transitionary state of sending out the same data until an ack is finally received?

Fast recovery avoids the time loss due to dropping to one.



❑ Both congestion avoidance and fast retransmit are congestion control methods. Do they both start from a slow start?

See above.

❑ Is fast retransmit used alongside Slow Start defined earlier, or is it an alternative?

It is a later improvement.

Fast Retransmit

- ❑ Optional – implemented in TCP Reno
  (Earlier version was TCP Tahoe)
- ❑ Duplicate Ack indicates a lost/out-of-order segment
- ❑ On receiving 3 duplicate acks (4th ack for the same segment):
  - ➢ Enter Fast Recovery mode
    - ❑ Retransmit missing segment
    - ❑ Set SSThresh=CWND/2
    - ❑ Set CWND=SSThresh+3 MSS **(Note: CWND is inflated)**
    - ❑ Every subsequent duplicate ack: CWND=CWND+1MSS
  - ➢ When a new ack (not a duplicate ack) is received
    - ❑ Exit fast recovery
    - ❑ Set CWND=SSTHRESH **(Note: CWND is deflated back**)

Student Questions

❑ What's the difference between fast recovery and fast retransmit?

Fast recovery is a part of the fast retransmit method.

❑ What are the reasons for triple duplicate acks?

Three subsequent segments make it, but a segment still needs to be received. This ensures that the segment has been lost, not just delayed.

❑ If a duplicate ack means the segment was lost, why do you wait until the 3rd duplicate ack to enter fast recovery mode?

A duplicate ack means that with a high probability, the segment was lost. It may still show up if delayed along the path.

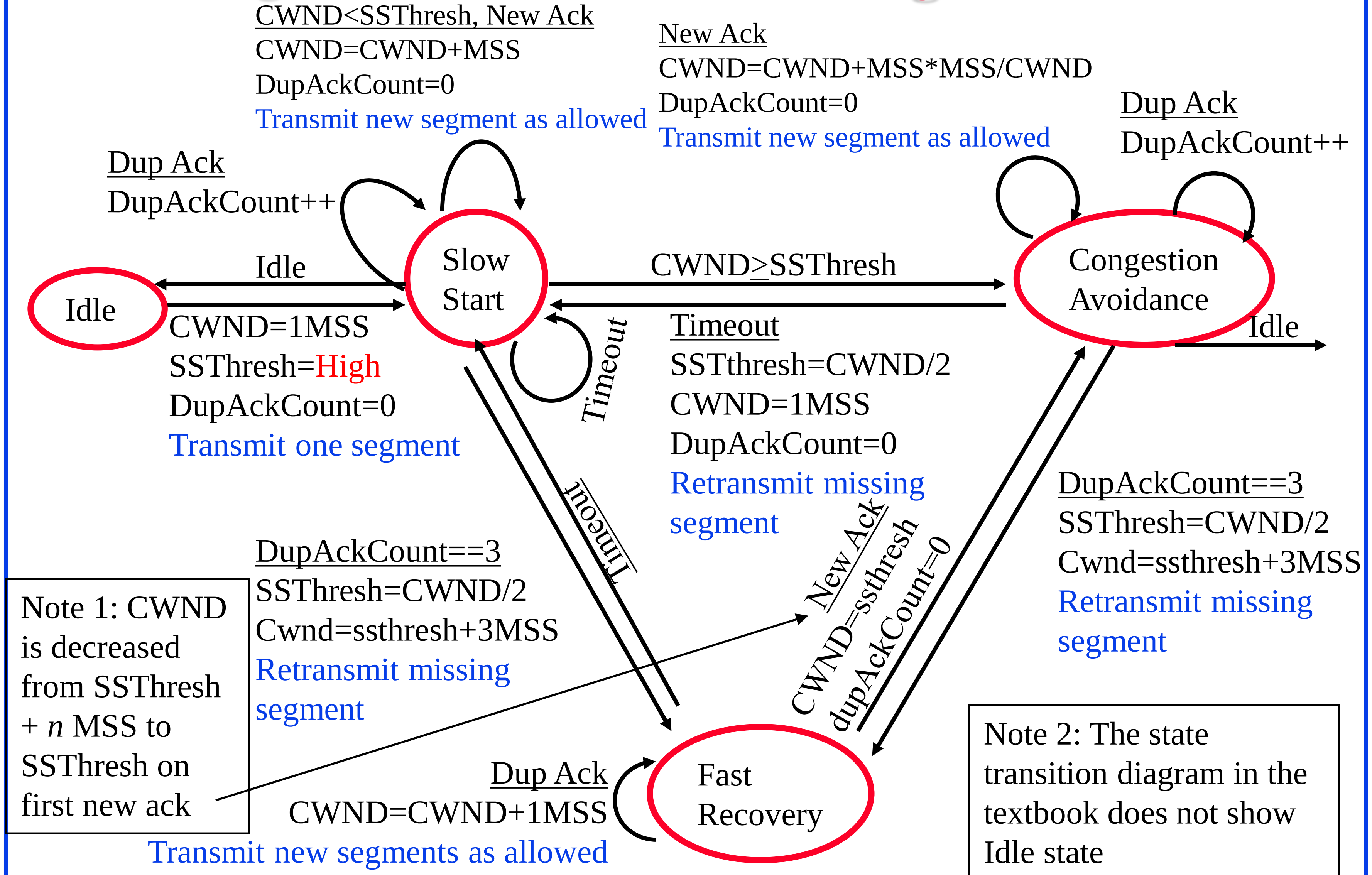
❖How does TCP Tahoe respond to triple duplicate ACK’s.

Book says it treats it as timeout. So the window goes to 1.

❖Why does TCP Reno respond differently to triple duplicate ACKs vs timeouts? Why is a response triggered for a triple duplicate ACK and not a double or quadruple duplicate ACK?

Simulation results showed that triple duplicate gives the best throughput.

TCP Congestion Control State Diagram



CWND<SSThresh, New Ack
CWND=CWND+MSS
DupAckCount=0
Transmit new segment as allowed

New Ack
CWND=CWND+MSS*MSS/CWND
DupAckCount=0
Transmit new segment as allowed

Dup Ack
DupAckCount++

Dup Ack
DupAckCount++

Idle

Slow Start

CWND≥SSThresh

Congestion Avoidance

Idle

Idle

CWND=1MSS
SSThresh=High
DupAckCount=0
Transmit one segment

Timeout

Timeout
SSTthresh=CWND/2
CWND=1MSS
DupAckCount=0
Retransmit missing segment

Timeout

New Ack
CWND=ssthresh
dupAckCount=0

DupAckCount==3
SSThresh=CWND/2
Cwnd=ssthresh+3MSS
Retransmit missing segment

DupAckCount==3
SSThresh=CWND/2
Cwnd=ssthresh+3MSS
Retransmit missing segment

Dup Ack
CWND=CWND+1MSS
Transmit new segments as allowed

Fast Recovery

Note 1: CWND is decreased from SSThresh + *n* MSS to SSThresh on first new ack

Note 2: The state transition diagram in the textbook does not show Idle state

Student Questions

- ❑ For the TCP Congestion Avoidance phase, for every new ACK, cwnd = cwnd + MSS·(MSS/cwnd). Graphs show the Congestion Avoidance phase to grow linearly, but is that a simplification? Because cwnd increases with every new ACK, the term MSS·(MSS/cwnd) shrinks with every ACK. Should the second term instead be MSS·(MSS/cwnd0), where cwnd0 is the initial value of cwnd when entering a round?

T=0 Window is W

T=1RTT W acks are received.
Increase by 1/W per ack
W=W+W/W = W+1

So the increase is linear in time.

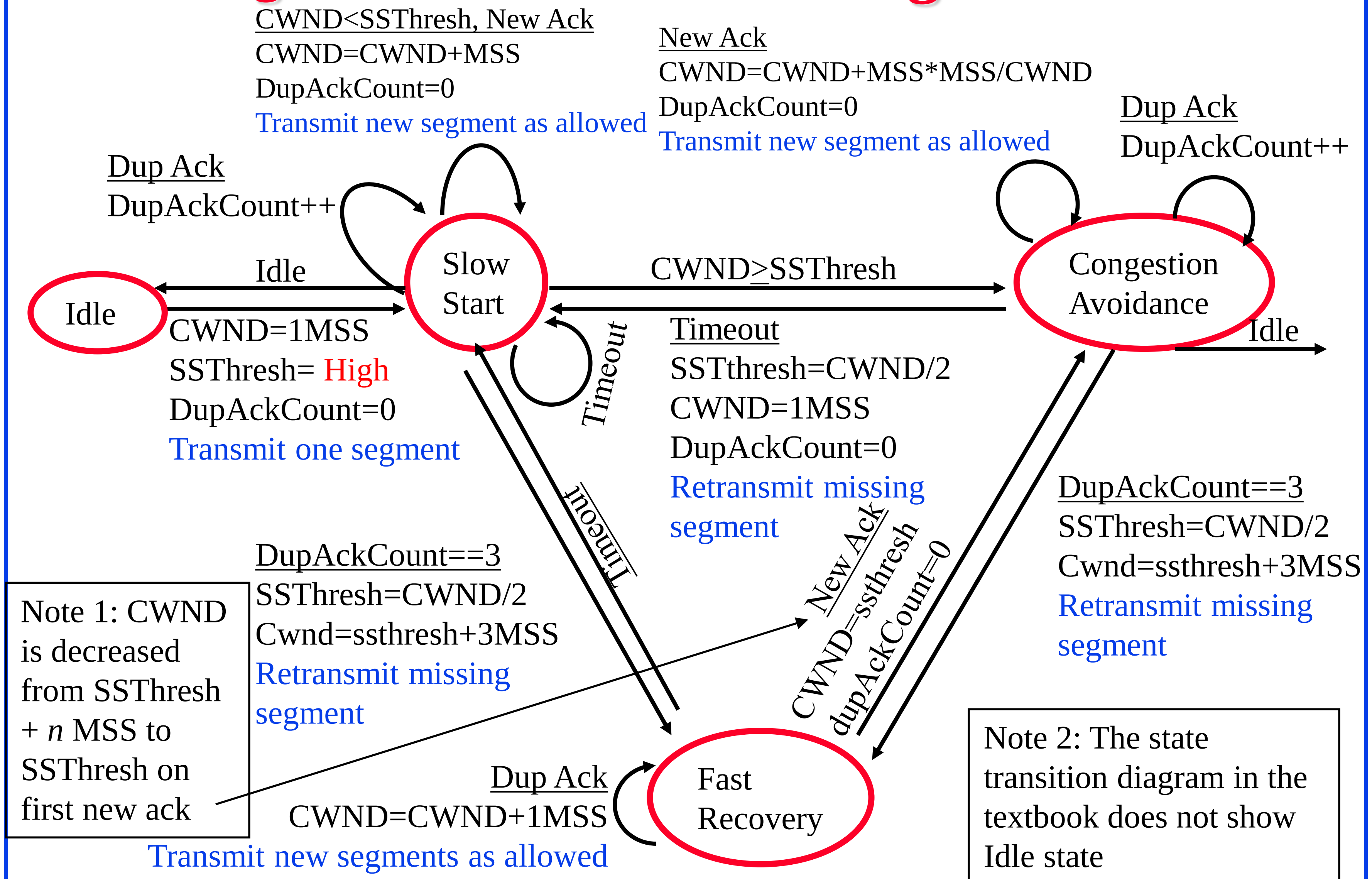
- ❑ Can you go back over this diagram one more time?

Sure.

- ❑ If the last transmission round during slow start is CWND=16 (so that the next round is CWND=32) and if SSThresh is a number like 20, will it switch to congestion control with CWND=20 or switch with CWND=32?

CWND is increased after each ack. So it will switch as soon as CWND goes over 20.

TCP Congestion Control State Diagram



CWND<SSThresh, New Ack
CWND=CWND+MSS
DupAckCount=0
Transmit new segment as allowed

New Ack
CWND=CWND+MSS*MSS/CWND
DupAckCount=0
Transmit new segment as allowed

Dup Ack
DupAckCount++

Dup Ack
DupAckCount++

Idle

Idle

Slow Start

CWND≥SSThresh

Congestion Avoidance

Idle

CWND=1MSS
SSThresh= High
DupAckCount=0
Transmit one segment

Timeout

Timeout
SSTthresh=CWND/2
CWND=1MSS
DupAckCount=0
Retransmit missing segment

Timeout

DupAckCount==3
SSThresh=CWND/2
Cwnd=ssthresh+3MSS
Retransmit missing segment

New Ack
CWND=ssthresh
dupAckCount=0

DupAckCount==3
SSThresh=CWND/2
Cwnd=ssthresh+3MSS
Retransmit missing segment

Note 1: CWND is decreased from SSThresh + *n* MSS to SSThresh on first new ack

Dup Ack
CWND=CWND+1MSS
Transmit new segments as allowed

Fast Recovery

Note 2: The state transition diagram in the textbook does not show Idle state

Student Questions

- ❑ Can we discuss the difference between a timeout and a triple duplicate ack again?

Timeouts are long. A number of "out-of-order" packets may reach the destination at that time. Each OoO packet is ack'ed with a "duplicate ack" and so triple duplicate ack indicates loss much sooner than the timeout.

- ❑ If a package gets lost at the very beginning (obviously in slow start) due to random reasons, then it will switch to congestion avoidance after a while; but if the line is spare, then CWND will increase at a very slow rate, and it loses the advantage of a slow start. Is it unavoidable?

Anything can be programmed, provided you can show that it is better. Actually, what is happening is the right thing to do. There is no need to avoid it.

- ❑ What is the difference between a timeout and a triple duplicate ack?

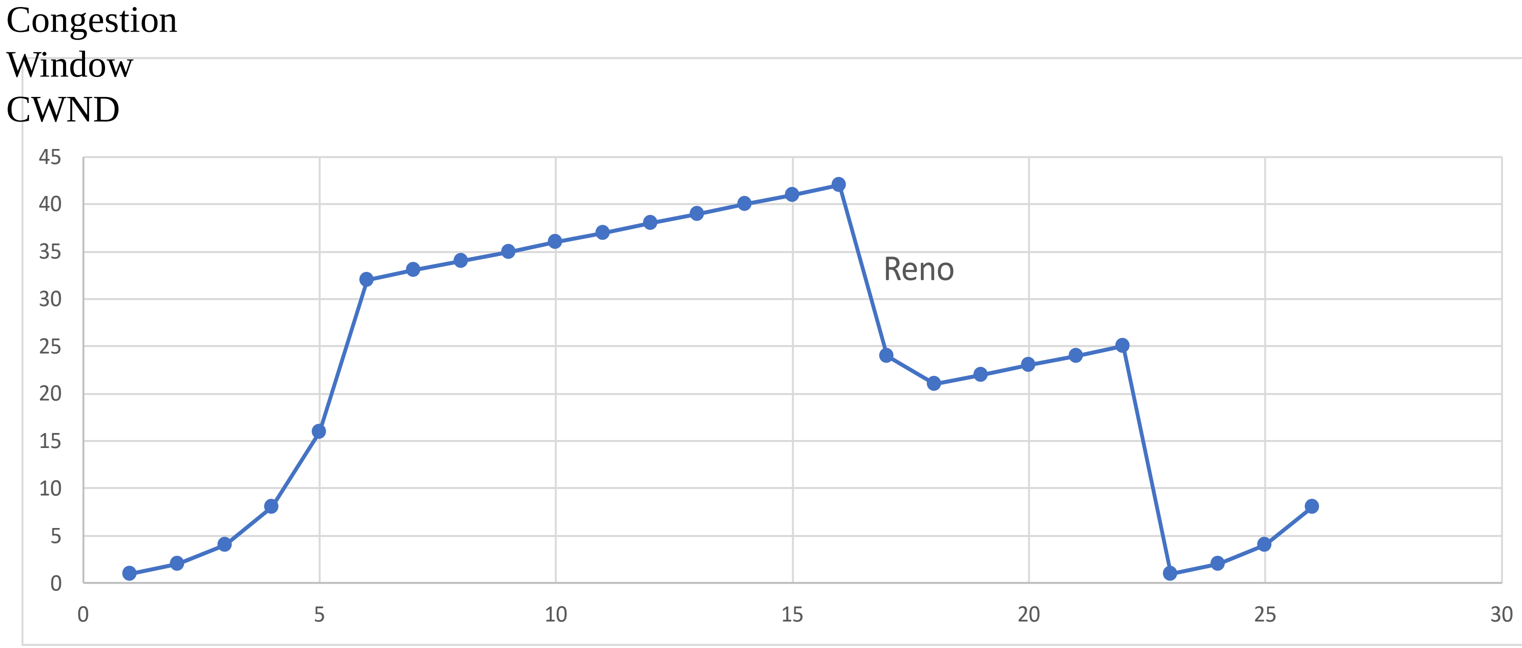
Timeout happens when a segment ack is not received within a reasonable time.

- ❖ Can you go over the TCP congestion control state diagram again?

Already done in the last TCP class.

Homework 3C: Slow Start

- [22 points] Consider the Figure below. Assuming TCP Reno is the protocol experiencing the behavior shown above, answer the following questions. In all cases, you should provide a short discussion justifying your answer.



	CWND
Round	Reno
1	1
2	2
3	4
4	8
5	16
6	32
7	33
8	34
9	35
10	36
11	37
12	38
13	39
14	40
15	41
16	42
17	24
18	21
19	22
20	23
21	24
22	25
23	1
24	2
25	4
26	8

Student Questions

- Could we go over the CWND graph problem?

Sure.

- What is the y-axis in the graph on HW 3C? MSS?

CWND in units of MSS.

Homework 3C (Cont)

- A. Identify the interval of time when TCP slow start is operating.
- B. Identify the intervals of time when TCP congestion avoidance is operating.
- C. After the 16th transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?
- D. After the 22nd transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?
- E. What is the initial value of ssthresh at the first transmission round?
- F . What is the value of ssthresh at the 18th transmission round?
- G. What is the value of ssthresh at the 24th transmission round?

Student Questions

- How to tell the difference between loss detected through timeout or duplicated ack?

Timeout happens when a timer expires. Duplicate ack happens when two acks are received. In the diagram, you distinguish whether there is a fast retransmit.

Homework 3C (Cont)

- H. During what transmission round is the 70th segment sent?
- I. Assuming a packet loss is detected after the 26th round by the receipt of a triple duplicate ACK, what will be the values of the congestion window size and of ssthresh?
- J. Suppose TCP Tahoe is used (instead of TCP Reno), and assume that triple duplicate ACKs are received at the 16th round. What are the ssthresh and the congestion window size at the 19th round? *(Hint: You need to calculate CWND in the 17-22nd rounds first. It will be different than that shown for Reno.)*
- K. Again, suppose TCP Tahoe is used, and there is a timeout event at the end of the 22nd round. How many packets have been sent out from the 17th round till the 22nd round, inclusive?

Student Questions

- ❑ Are we able to select the interval (idle interval) after which the connection breaks? *Yes. Applications set it according to their needs.*
- ❑ Why SSThresh to CWND/2 rather than CWND? *You need to reduce the load (window) during congestion.*
- ❖ **For clarification, If CWND is 15, would SSThresh be 7 or 8?** *In the code, CWND is maintained in Bytes, so it is possible for it to be 7.5 MSS. However, when in this situation, the sender should not send a small segment of size 0.5MSS. This leads to issues. So the effective window size will be 7. Ref: V. Jacobson and M. Karels, "Congestion Avoidance and Control" Nov. 1988,* https://ee.lbl.gov/papers/congavoid.pdf *This is a very slightly revised version of their SIGCOMM'88 paper.*

TCP Average Throughput

- Average Throughput = $\dfrac{1.22\ \text{MSS}}{\text{RTT}\ \sqrt{P}}$
- Here, P = Probability of Packet loss.
- Note 1: The formula is an approximation that does not apply at P=0 or P=1. At P=1, the throughput is zero. At P=0, the throughput is min{1, (Receiver Window/RTT)}
- Note 2: The textbook has a different formula. Numerous such formulas are in literature. All under different assumptions and some empirical ones. This formula is not exact or universally agreed.

Student Questions

- So this equation only works when P <= 1%?

Yes, it is an approximation that works at low non-zero values of P.

- As for the optional homework 3D, did we suppose to apply this average throughput formula for solving that question?

No. That homework is similar to a slow start. You need to determine the window at each roundtrip and count the number of packets.

- Is there a reason why we assume that packet loss cannot be greater than 10%?

At 10%, every 10th packet will be lost, and the network will seem very slow or broken.

- When the window lengths are different, whether the different TCP connections are fair or not.

The user throughput depends upon the window size and RTT. Most congestion schemes are tested to check for fairness at the bottleneck, even if RTTs are different. => Large windows for Large RTTs

TCP Average Throughput

- Average Throughput = $$\frac{1.22\ \text{MSS}}{\text{RTT}\ \sqrt{P}}$$
- Here, P = Probability of Packet loss.
- Note 1: The formula is an approximation that does not apply at P=0 or P=1. At P=1, the throughput is zero. At P=0, the throughput is min{1, (Receiver Window/RTT)}
- Note 2: The textbook has a different formula. Numerous such formulas are in literature. All under different assumptions and some empirical ones. This formula is not exact or universally agreed.

Student Questions

- Where does "1.22" come from?

This is curve-fitting.

- Where does the 1.22 constant come from for the average throughput?

1.22=Sqrt(1.5)

*See:*http://www.cs.emory.edu/~cheung/Courses/558/Syllabus/07-TCP-Anal/padhye-2.html

- Why there's a square root for P?

See the above reference.

- Also, the book has an equation for TCP Reno that is (.75*W)/RTT.

That assumes zero timeouts. Duplicate acks detect all drops.

- P would be smaller than 1%?

In that range.

- I assume P=0 means 0% chance of packet loss while one is guaranteed packet loss:?

Yes.

TCP Average Throughput

- Average Throughput = $\dfrac{1.22\ \text{MSS}}{\text{RTT}\ \sqrt{P}}$
- Here, P = Probability of Packet loss.
- Note 1: The formula is an approximation that does not apply at P=0 or P=1. At P=1, the throughput is zero. At P=0, the throughput is min{1, (Receiver Window/RTT)}
- Note 2: The textbook has a different formula. Numerous such formulas are in literature. All under different assumptions and some empirical ones. This formula is not exact or universally agreed.

Student Questions

- Please go over what the control bits on slide 33 do again.

Sure.

- Can you go over the TCP Congestion Control State Diagram again?

Sure.

- Does the probability of loss vary with different types of wireless connections? (Wi-Fi vs. LTE vs. 4G vs. 5G, etc.)

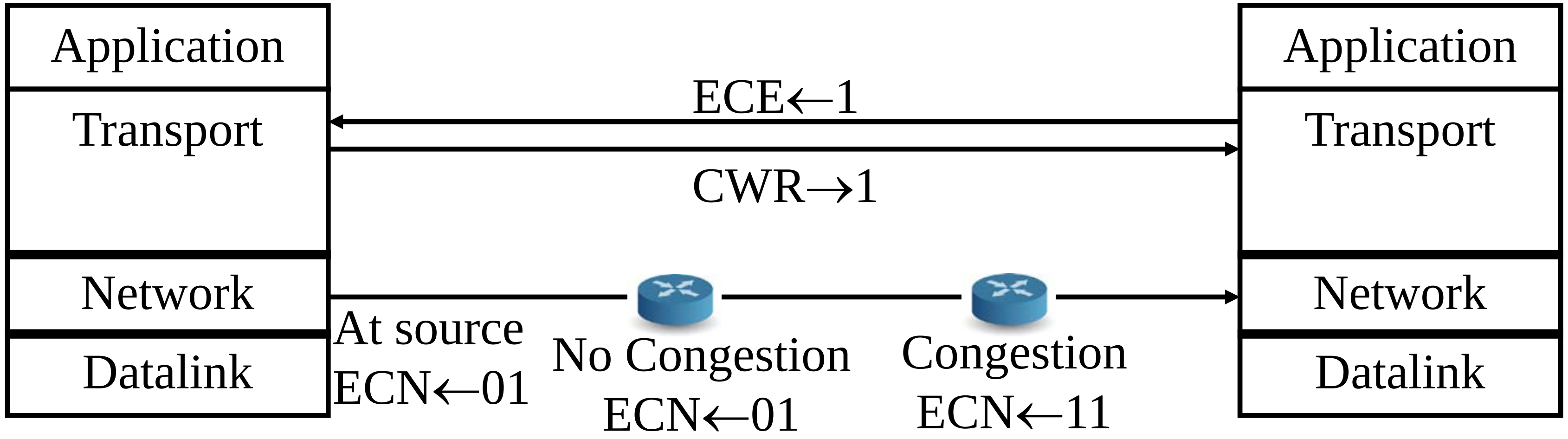
Yes, more than the technologies, the wireless performance depends on the environment.

- What/Who decides on what source port to send? Is it OS? Can we decide on that or the application/browser? Can we decide on what flow control or congestion control is being used, or are they all set up under the hood?

You, the user, can decide and indicate the port #. Flow control and congestion control are decided by the network protocol implementers.

Explicit Congestion Notification (ECN)

- ❑ Explicit congestion notification (ECN) is based on our DECbit research.
  - ➢ Two bits in IP Header: Last two bits of traffic class (Next chapter)
  - ➢ Two bits in the TCP header: ECE and CWR
- ❑ IP Bits:
  - ➢ 00: Transport is not capable of ECN (e.g., UDP)
  - ➢ 01 or 10: ECN capable transport
  - ➢ 11: Congestion Experienced
- ❑ When a router encounters congestion, instead of dropping the datagram, it marks the two bits as "11" congestion experienced.



Application | Transport | Network | Datalink

ECE←1

CWR→1

At source ECN←01

No Congestion ECN←01

Congestion ECN←11

Application | Transport | Network | Datalink

Student Questions

- ❑ Is it just the IP / TCP header getting sent back? Or is the whole packet itself getting sent back?

The bits are set in the TCP segments going in the reverse direction on the same TCP connection.

- ❑ How many packets in a row does it take before the router starts marking new ones as '11'?

If the average queue length>1. The routers are required to compute the average queue length. They can do this by exponentially weighted average.

- ❑ Are the ECN bits set in the Transport layer header or the Network layer header?

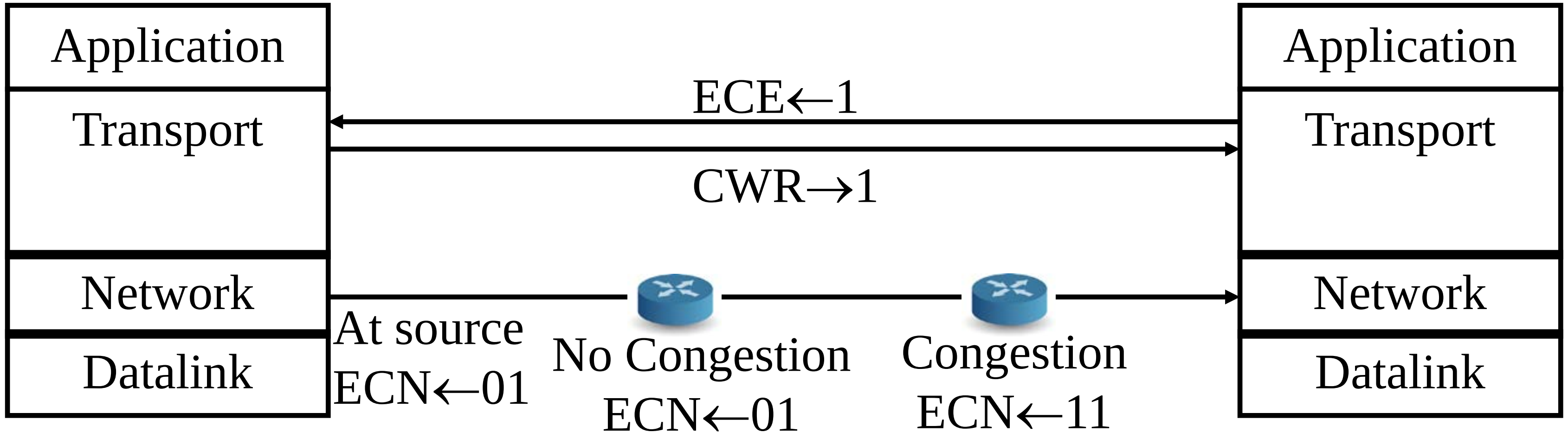
Both

- ❑ Can you please explain again how RTT and ECN are related? With a longer RTT and a congested network, would you not be sacrificing performance?

No. We are not sacrificing performance. We are maximizing performance. All of this was analyzed, measured, and simulated before publishing the scheme.

Explicit Congestion Notification (ECN)

- Explicit congestion notification (ECN) is based on our DECbit research.
  - Two bits in IP Header: Last two bits of traffic class (Next chapter)
  - Two bits in the TCP header: ECE and CWR
- IP Bits:
  - 00: Transport is not capable of ECN (e.g., UDP)
  - 01 or 10: ECN capable transport
  - 11: Congestion Experienced
- When a router encounters congestion, instead of dropping the datagram, it marks the two bits as “11” congestion experienced.



Application

Transport

Network

Datalink

ECE←1

CWR→1

At source ECN←01

No Congestion ECN←01

Congestion ECN←11

Application

Transport

Network

Datalink

Student Questions

- Is congestion declared when the buffer of the router is overflowed or about to overflow?

In ECN, congestion is declared when the average queue length is more than 1.

- Could you briefly review ECN (Slide 3-51)? *Sure.*
- What's the difference between ECN and CE? I noticed they both send from source to destination.

ECN is the name of the field. CE is one possible value for that field.

- Can you clarify the difference between 01 and 10?

Routers treat both as the same.

- Why does ECN use the IP header instead of the protocol header?

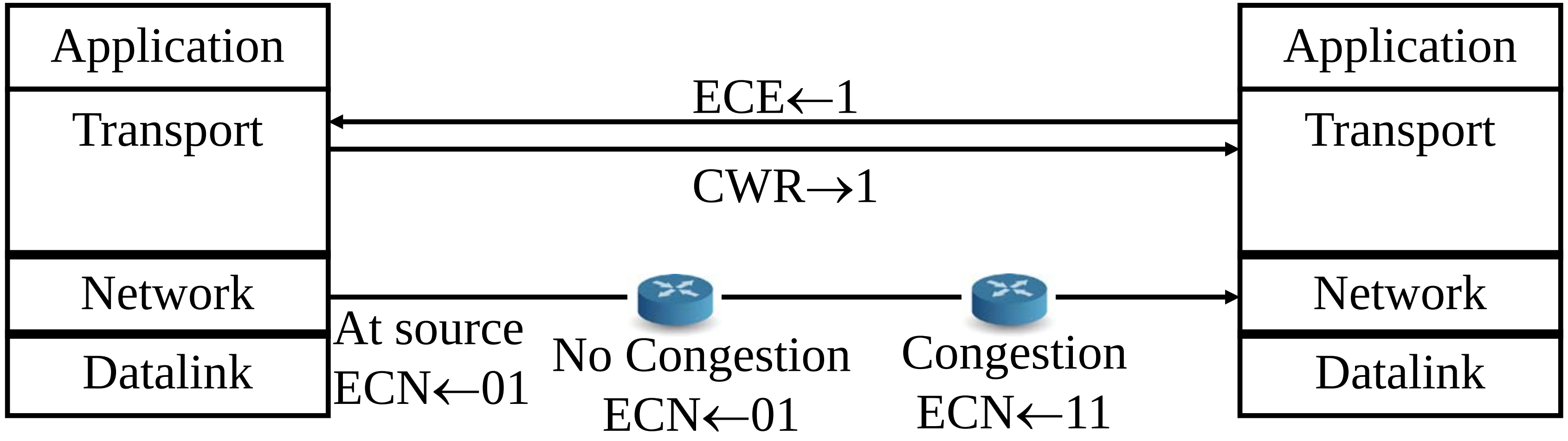
It uses both IP and TCP headers.

- Should we view the two-bit collectively, or does each have its meaning?

ECN is two bits collectively.
ECE and CWR are individual bits.

Explicit Congestion Notification (ECN)

- Explicit congestion notification (ECN) is based on our DECbit research.
  - Two bits in IP Header: Last two bits of traffic class (Next chapter)
  - Two bits in the TCP header: ECE and CWR
- IP Bits:
  - 00: Transport is not capable of ECN (e.g., UDP)
  - 01 or 10: ECN capable transport
  - 11: Congestion Experienced
- When a router encounters congestion, instead of dropping the datagram, it marks the two bits as "11" congestion experienced.



Application | Transport | Network | Datalink

ECE←1

CWR→1

At source
ECN←01

No Congestion
ECN←01

Congestion
ECN←11

Application | Transport | Network | Datalink

Student Questions

- Is there a code for if there isn't congestion experienced?

01 or 10
It can indicate two kinds of transport, but routers do not distinguish these two types.

- So the router will set all the errors to 11, no matter if it is 01 or 10?

Yes. All routers experiencing congestion will set ECN on all packets (not errors)

- How can these ECN messages be sent? Are all messages cached on all nodes of the route, so if one times out, that node can send ECN back? That seems like many resources.

ECN are bits in the headers of the packets passing by. These are not messages. They are not stored anywhere. They are only set and counted at the destination.

- Can you explain why ECN from the source is 11 here and what that means?

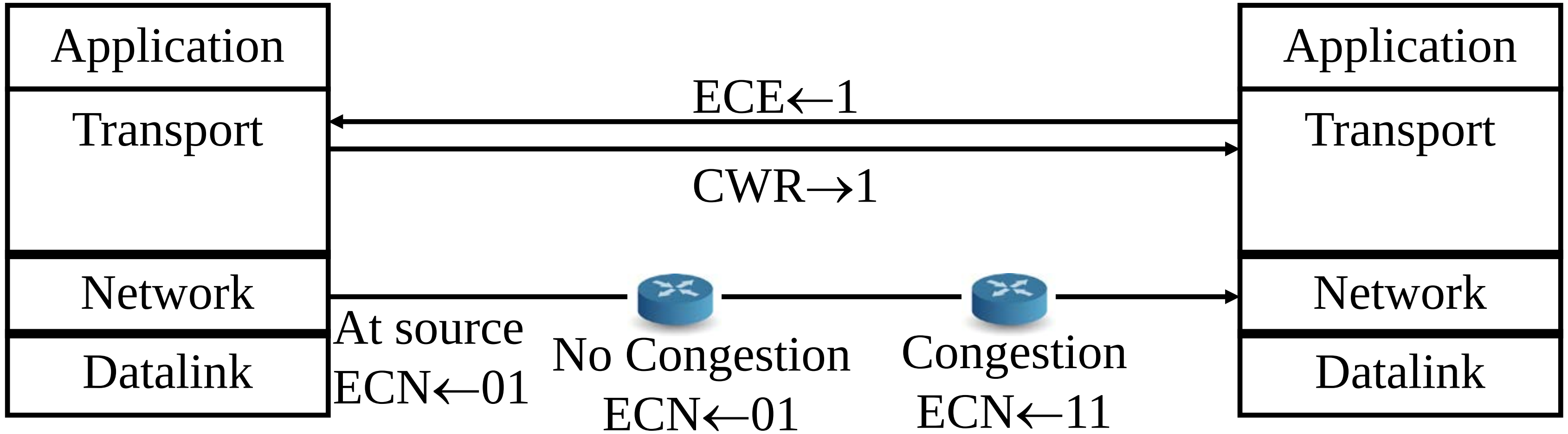
ECN from sources is 00, 01, or 10. It is set to 11 by routers.

Explicit Congestion Notification (ECN)

- Explicit congestion notification (ECN) is based on our DECbit research.
  - Two bits in IP Header: Last two bits of traffic class (Next chapter)
  - Two bits in the TCP header: ECE and CWR
- IP Bits:
  - 00: Transport is not capable of ECN (e.g., UDP)
  - 01 or 10: ECN capable transport
  - 11: Congestion Experienced

TCP Bits:
01: Reset cong Window
10: Cong window reset
11: Both

- When a router encounters congestion, instead of dropping the datagram, it marks the two bits as "11" congestion experienced.



Application
Transport
Network
Datalink

ECE←1
CWR→1

At source
ECN←01

No Congestion
ECN←01

Congestion
ECN←11

Application
Transport
Network
Datalink

Student Questions

- Why ECN elects to use 2 bits instead of 1?

There are three possibilities in both IP and TCP.

- Why should it distinguish between ECN-incapable and ECN-capable transports?

Because it is a later improvement, older nodes will not have it.

- What does ECN mean? What information does it give?

ECN⇒I am congested!

- Why is UDP not capable of ECN? Is it just too old and can not implement such technology?

It does not have flow control or congestion control.

- What does CWR stand for?

Congestion Window Reset

- Is this why TCP and IP rely on each other, or are there other occurrences of the layered packet structure being broken between TCP/IP?

Yes. We will see during IP.

- What is the role of routers in this schema?

Routers implement the Network layer.

- What makes ECN so important?

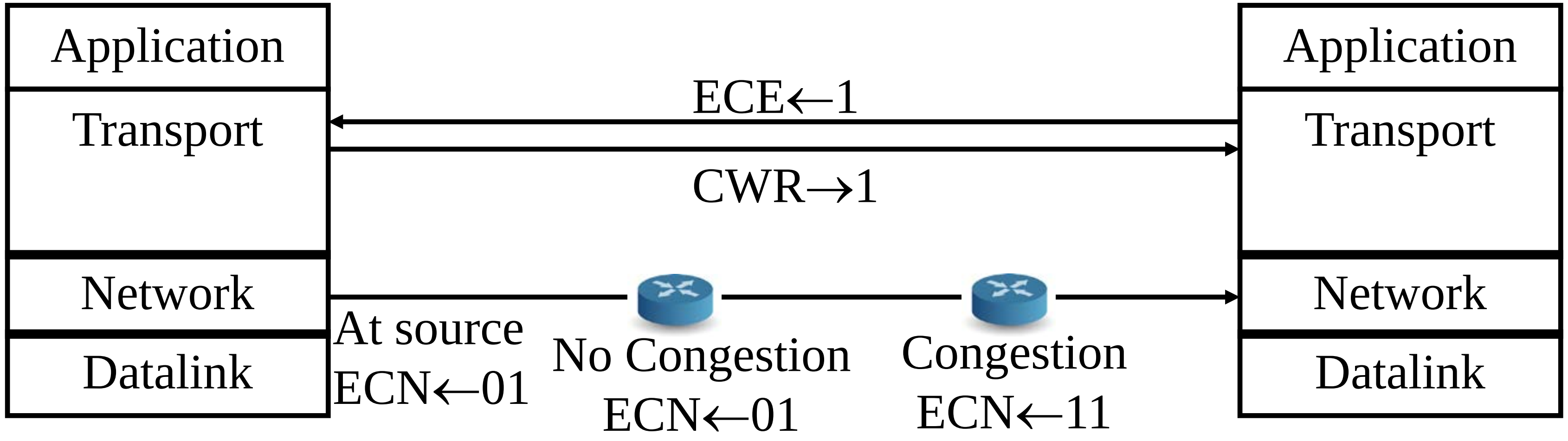
Lossless congestion control

- What exactly is a DECbit?

DECbit is the name of our scheme, which we developed when we were working at Digital Equipment Corporation (DEC). My co-author presented this to IETF, and they called it Explicit Congestion Notification (ECN).

ECN (Cont)

- ECN uses two bits in the TCP header: ECE and CWR
- On receiving "CE" code point, the receiver sends "ECN Echo (ECE)" flag in the TCP header
- On seeing the ECE flag, the source reduces its congestion window, and sets "Congestion Window Reduced (CWR) flag in the outgoing segment
- On receiving "CWR" flag, the receiver, stops setting ECE bit



Student Questions

- What does CRW mean?

CRW="I heard you. I have reduced the window."

- Is this just another protocol to manage congestion? Do all TCP protocols have different ways to manage congestion?

All routers implement all schemes. Some schemes are not implemented by some routers, and so the schemes have to allow for such routers.

- ECN marks the two bits such as 00,01,10,11. Is the above situation happen at the same time?

No. In one instance, only one marking is used.

- In the diagram, why are the "ECE" and "CWR" being transmitted? I thought it only gets transmitted during "CE."“

ECE and CWR are always there but maybe 0.

- Since ECN involves routers, is it in the network layer?

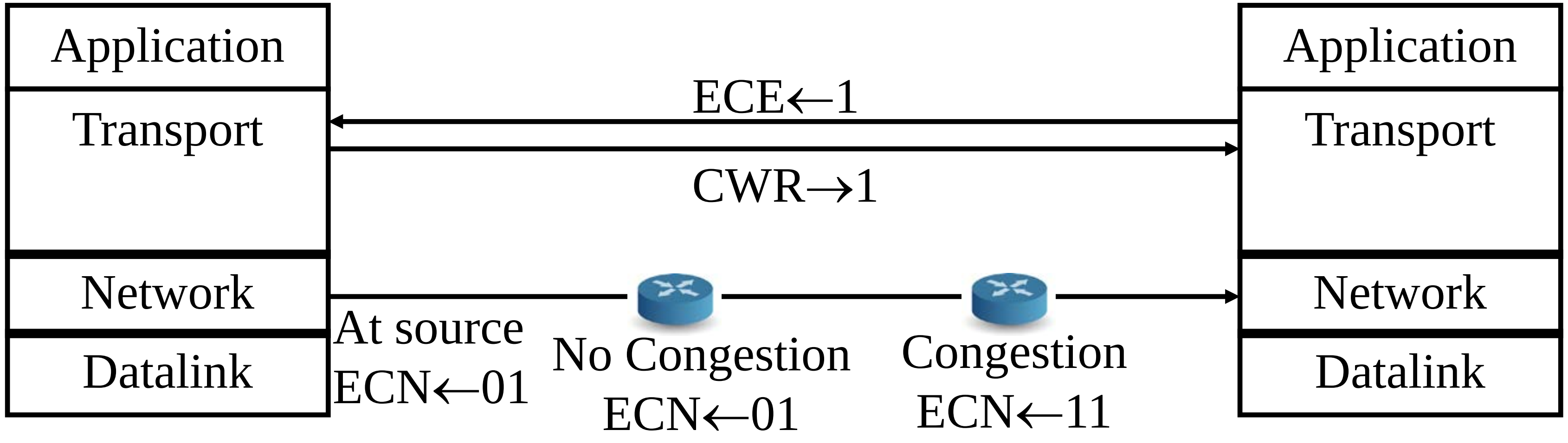
Yes.

Why does the IP header contain ECE and CWR codes if the TCP bits already include those?

Slide corrected.

ECN (Cont)

- ECN uses two bits in the TCP header: ECE and CWR
- On receiving "CE" code point, the receiver sends "ECN Echo (ECE)" flag in the TCP header.
- On seeing the ECE flag, the source reduces its congestion window, and sets "Congestion Window Reduced (CWR) flag in the outgoing segment
- On receiving "CWR" flag, the receiver, stops setting ECE bit



Student Questions

- How is congestion control different from flow control?

Flow control is the control by the receiver. Congestion control is by the network.

- When does the "CWR" flag stop being set?

When the congestion window is reduced, it keeps setting it to 1 until the ECE bit reaches zero.

- Can you explain the main difference between the bits in the IP header vs. the TCP header again? *Sure.*
- When a router receives ECN = 10, why it will deliver an MSS with ECN =10 and then deliver a new MSS with ECN = 11? Why not just deliver a MSS wt ECN = 11?

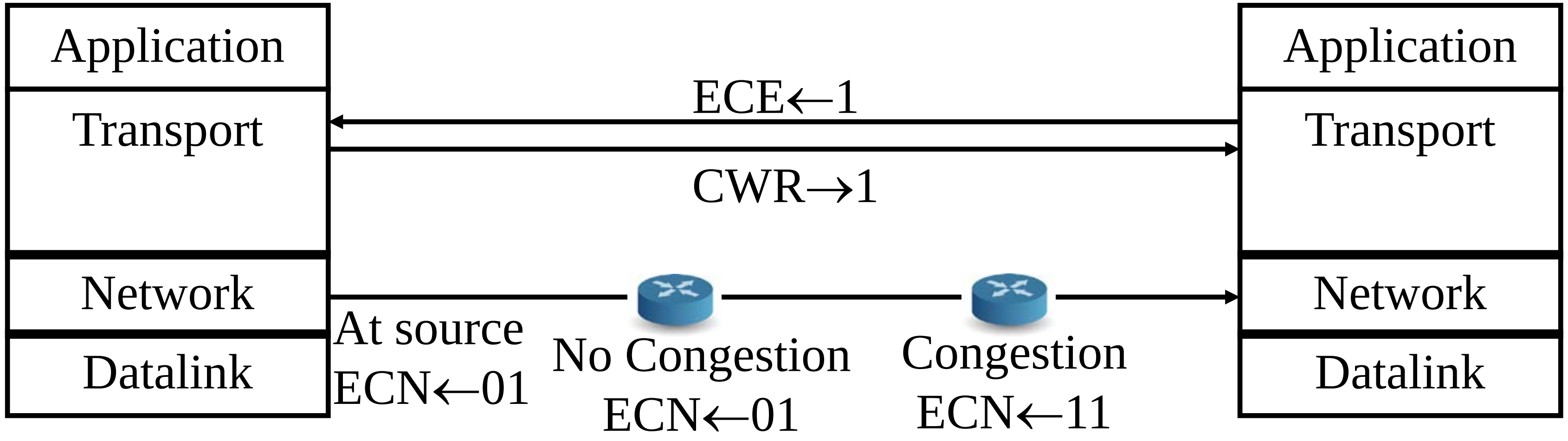
MSS = Maximum segment size
Segment=TCP PDU
Datagram=IP PDU
When a router received ECN=10 it changes it to 11 if and only if it is congested.

- What is the purpose of having two sets of bits to signal congestion, one in the IP header and the other in the TCP header?

IP sets. TCP reacts.

ECN (Cont)

- ECN uses two bits in the TCP header: ECE and CWR
- On receiving "CE" code point, the receiver sends "ECN Echo (ECE)" flag in the TCP header.
- On seeing the ECE flag, the source reduces its congestion window, and sets "Congestion Window Reduced (CWR) flag in the outgoing segment
- On receiving "CWR" flag, the receiver, stops setting ECE bit



Student Questions

- Is it possible for a receiver to receive duplicate ECE codes after it has already reduced and sent its CWR code?

Yes. ECE=1 bits are sent continuously until CWR=1 is received. CWR=1 is sent continuously until ECE=0 is received.

- How is the explicit congestion notification ECN option utilized in TCP?

TCP Reduces its window when the network is congested, and ECN is received.

- How are ECN and ECE/CWR both independent and work together?

They are NOT independent. The two layers work together to avoid congestion.

- What is "CE code point" meant, and how does it connect to receiver-side congestion detection?

CE=11 is received at the receiver

- Is ECN dictating the congestion window?

Yes

TCP: Summary

1. TCP uses **port numbers** for multiplexing
2. TCP provides reliable **full-duplex** connections.
3. TCP is **stream** based and has **window flow control**
4. **Slow-start congestion control** works on timeout
5. **Explicit congestion notification** works using ECN bits

Please see Slide 3.69 for discussion on CUBIC.

Read Sections 3.5, 3.6, and 3.7. Do R14-R19, P25, P26, P31, P32, P33, P40

Student Questions

- From Book Page 252: TCP has similarities with GBN/selective repeat but does not explicitly use these, correct?

Selective ack has now been added to TCP. Particularly helpful for highspeed networks. In SA, you can list the missing segments. So better than both GBN and SR.

- How does IP realize the congestion?

By watching its queues.

- Why is ECN not possible if both bits are set, or neither are set?

ECN is always possible. If both bits are clear, there is no congestion. If both bits are 1, there is congestion.

- Could you briefly explain again what it means to reduce the congestion window?

Multiplicative decrease by a factor.

- What if the Header field for ECN is corrupted and sends the fake information to the source?

The source will reduce the window.

- Does TCP have any way to fight against UDP hogging bandwidth?

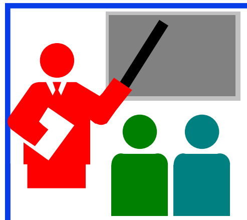
Indirectly. The queue length includes both kinds of packets.

- How would the network know if the congestion has been resolved? What flags would be set?

CE, ECE, and CRA will be zero.

- Why explicit congestion use ECN?

ECN=Explicit Congestion Notification



TCP: Summary

1. TCP uses **port numbers** for multiplexing
2. TCP provides reliable **full-duplex** connections.
3. TCP is **stream** based and has **window flow control**
4. **Slow-start congestion control** works on timeout
5. **Explicit congestion notification** works using ECN bits

Please see Slide 3.69 for discussion on CUBIC.

Read Sections 3.5, 3.6, and 3.7. Do R14-R19, P25, P26, P31, P32, P33, P40

Student Questions

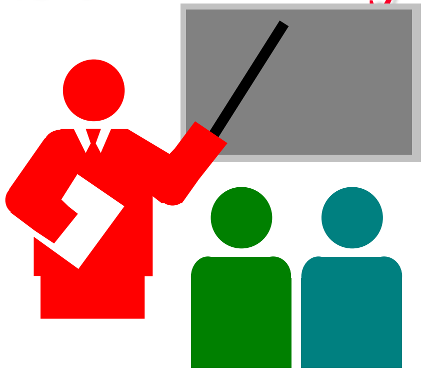
- ❑ I saw that you have a lot of papers on recent breakthroughs for blockchain, do web3 and blockchain use TCP as well? Or does it use other protocols?

Web3 uses TCP. Blockchains UDP for broadcasts. TCP for queries.

- ❑ What are examples of full-duplex connections, and what is the alternative to a full-duplex connection?

All TCP connections are full duplex. A client sends a query to the server and server sends a response to the client. This is full duplex.

Summary



1. **Multiplexing/demultiplexing** by a combination of source and destination IP addresses and port numbers.
2. **Longer distance or higher speed**
   ⇒ A larger α ⇒ Larger window is better.
3. Window flow control is better for long-distance or high-speed networks
4. UDP is connectionless and simple.
   No flow/error control. Has error **detection**.
5. TCP provides **full-duplex** connections with flow/error/**congestion** control.

Student Questions

- ❑ Shouldn't TCP be using both port numbers and IP addresses for multiplexing?

IP takes care of the IP address. TCP takes care of port #.

- ❑ Can we say that TCP is a layer of abstraction on top of UDP?

You can implement it that way, but most people think of them as side-by-side and not one on top of another.

- ❑ I wonder whether application layer protocols such as HTTP also use IP addresses and port numbers for multiplexing.

Yes. Almost all application protocols.

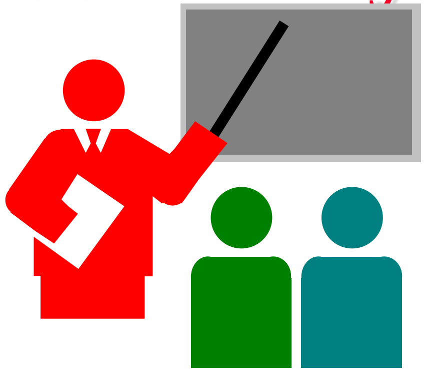
- ❑ Once we detect an error in a packet when using UDP, do we just discard the packet?

Yes.

- ❑ Could you reiterate what the difference is between flow and congestion control? Why do they need to be two distinct concepts?

Flow control takes care of buffers at the destination. Congestion control takes care of the buffers in the routers.

Summary



1. **Multiplexing/demultiplexing** by a combination of source and destination IP addresses and port numbers.
2. **Longer distance or higher speed**
   ⇒ A larger α ⇒ Larger window is better.
3. Window flow control is better for long-distance or high-speed networks
4. UDP is connectionless and simple.
   No flow/error control. Has error **detection**.
5. TCP provides **full-duplex** connections with flow/error/**congestion** control.

Student Questions

- ❑ How does TCP adapt to long-distance networks compared to UDP?

Long distance networks need large windows. UDP does not have windows.

Lab 3: Reliable Transport Protocol

[60 points] Overview

In this laboratory programming assignment, you will be writing the sending and receiving transport-level code for implementing a simple reliable data transfer protocol. There are two versions of this lab, the Alternating-Bit-Protocol version and the Go-Back-N version. This lab should be **fun** since your implementation will differ very little from what would be required in a real-world situation.

Since you probably don't have standalone machines (with an OS that you can modify), your code will have to execute in a simulated hardware/software environment. However, the programming interface provided to your routines, i.e., the code that would call your entities from above and from below is very close to what is done in an actual UNIX environment. (Indeed, the software interfaces described in this programming assignment are much more realistic that the infinite loop senders and receivers that many texts describe). Stopping/starting of timers are also simulated, and timer interrupts will cause your timer handling routine to be activated.

The routines you will write

The procedures you will write are for the sending entity (A) and the receiving entity (B). Only unidirectional transfer of data (from A to B) is required. Of course, the B side will have to send packets to A to acknowledge (positively or negatively) receipt of data. Your routines are to be implemented in the form of the procedures described below. These procedures will be called by (and will call) procedures that I have written which emulate a network environment. The overall structure of the environment is shown in Figure Lab.3-1 (structure of the emulated environment):

The unit of data passed between the upper layers and your protocols is a *message*, which is declared as:

```
struct msg { char data[20];
};
```

This declaration, and all other data structure and emulator routines, as well as stub routines (i.e., those you are to complete) are in the file, **prog2.c (http://gaia.cs.umass.edu/kurose/transport/prog2.c).** Your sending entity will thus receive data in 20-byte chunks from layer5; your receiving entity should deliver 20-byte chunks of correctly received data to layer5 at the receiving side.

Student Questions

❑ Is the deadline for Lab3 on 16th or 14th?

16th.

❑ The alternating bits version is described, but the Go-Back-N version is not. Can you explain it a bit?

Go-back-N was described in Slide 3-21.

❑ Where are the instructions for the go-back N version?

Removed to reduce student load.

Could you elaborate more about the implementation of the GBN version?

❑ Is the design document required for this lab? *No.*

❑ Is the lab also written in Python?

Most of the system software uses C. Python is used for applications. TCP is a part of the operating systems.

Lab 3 (Cont)

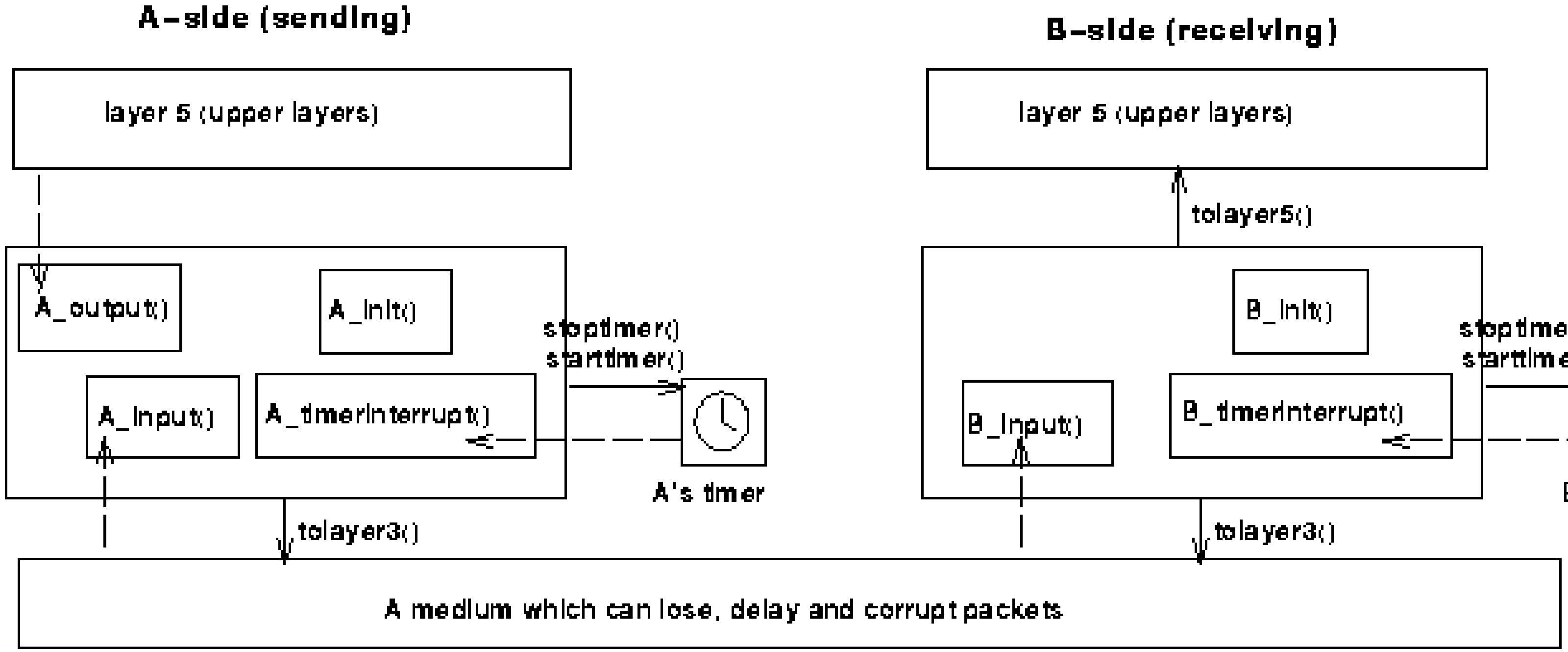


Figure Lab.3-1

Student Questions

- ❑ When we do lab 3 for GBN, if the packet received by the receiver from the application layer is outside the window, or meets buffer overflow issues, how to fix it?

There is no flow control on the traffic coming from the application to TCP. Packets can be lost due to buffer overflow. Inter-process communication in the OS may prevent the application from proceeding since this is local.

Lab 3 (Cont)

The unit of data passed between your routines and the network layer is the *packet*, which is declared as:

```
struct pkt { int seqnum; int acknum;
int checksum; char payload[20];
};
```

Your routines will fill in the payload field from the message data passed down from layer5. The other packet fields will be used by your protocols to insure reliable delivery, as we've seen in class.

The routines you will write are detailed below. As noted above, such procedures in real-life would be part of the operating system, and would be called by other procedures in the operating system.

A_output(message), where message is a structure of type msg, containing data to be sent to the B-side. This routine will be called whenever the upper layer at the sending side (A) has a message to send. It is the job of your protocol to insure that the data in such a message is delivered in-order, and correctly, to the receiving side upper layer.

A_input(packet), where packet is a structure of type pkt. This routine will be called whenever a packet sent from the B-side (i.e., as a result of a tolayer3() being done by a B-side procedure) arrives at the A-side. packet is the (possibly corrupted) packet sent from the B-side.

A_timerinterrupt() This routine will be called when A's timer expires (thus generating a timer interrupt). You'll probably want to use this routine to control the retransmission of packets. See starttimer() and stoptimer() below for how the timer is started and stopped.

A_init() This routine will be called once, before any of your other A-side routines are called. It can be used to do any required initialization.

B_input(packet),where packet is a structure of type pkt. This routine will be called whenever a packet sent from the A-side (i.e., as a result of a tolayer3() being done by a A-side procedure) arrives at the B-side. packet is the (possibly corrupted) packet sent from the A-side.

B_init() This routine will be called once, before any of your other B-side routines are called. It can be used to do any required initialization.

Student Questions

❑ For lab 3, do we have to worry about B's timer?

No. Receivers do not have timers.

❑ Can we have an additional office hour for the lab due on Wednesday this week? Also, will there be any additional office hours before the exam?

I will check with our TAs. Additional hours, if any, will be announced on Piazza.

Lab 3 (Cont)

Software Interfaces

The procedures described above are the ones that you will write. I have written the following routines which can be called by your routines:

starttimer(calling_entity,increment), where calling_entity is either 0 (for starting the A-side timer) or 1 (for starting the B side timer), and increment is a *float* value indicating the amount of time that will pass before the timer interrupts. A's timer should only be started (or stopped) by A-side routines, and similarly for the B-side timer. To give you an idea of the appropriate increment value to use: a packet sent into the network takes an average of 5 time units to arrive at the other side when there are no other messages in the medium.

stoptimer(calling_entity), where calling_entity is either 0 (for stopping the A-side timer) or 1 (for stopping the B side timer).

tolayer3(calling_entity,packet), where calling_entity is either 0 (for the A-side send) or 1 (for the B side send), and packet is a structure of type pkt. Calling this routine will cause the packet to be sent into the network, destined for the other entity.

tolayer5(calling_entity,message), where calling_entity is either 0 (for A-side delivery to layer 5) or 1 (for B-side delivery to layer 5), and message is a structure of type msg. With unidirectional data transfer, you would only be calling this with calling_entity equal to 1 (delivery to the B-side). Calling this routine will cause data to be passed up to layer 5.

Student Questions

Lab 3 (Cont)

The simulated network environment

A call to procedure tolayer3() sends packets into the medium (i.e., into the network layer). Your procedures A_input() and B_input() are called when a packet is to be delivered from the medium to your protocol layer.

The medium is capable of corrupting and losing packets. It will not reorder packets. When you compile your procedures and my procedures together and run the resulting program, you will be asked to specify values regarding the simulated network environment:

Number of messages to simulate. My emulator (and your routines) will stop as soon as this number of messages have been passed down from layer 5, regardless of whether or not all of the messages have been correctly delivered. Thus, you need **not** worry about undelivered or unACK'ed messages still in your sender when the emulator stops. Note that if you set this value to 1, your program will terminate immediately, before the message is delivered to the other side. Thus, this value should always be greater than 1.

Loss. You are asked to specify a packet loss probability. A value of 0.1 would mean that one in ten packets (on average) are lost.

Corruption. You are asked to specify a packet loss probability. A value of 0.2 would mean that one in five packets (on average) are corrupted. Note that the contents of payload, sequence, ack, or checksum fields can be corrupted. Your checksum should thus include the data, sequence, and ack fields.

Tracing. Setting a tracing value of 1 or 2 will print out useful information about what is going on inside the emulation (e.g., what's happening to packets and timers). A tracing value of 0 will turn this off. A tracing value greater than 2 will display all sorts of odd messages that are for my own emulator-debugging purposes. A tracing value of 2 may be helpful to you in debugging your code. You should keep in mind that *real* implementors do not have underlying networks that provide such nice information about what is going to happen to their packets!

Average time between messages from sender's layer5. You can set this value to any non-zero, positive value. Note that the smaller the value you choose, the faster packets will be be arriving to your sender.

Student Questions

Lab 3 (Cont)

The Alternating-Bit-Protocol Version of this lab.

You are to write the procedures, A_output(),A_input(),A_timerinterrupt(),A_init(),B_input(), and B_init() which together will implement a stop-and-wait (i.e., the alternating bit protocol, which we referred to as rdt3.0 in the text) unidirectional transfer of data from the A-side to the B-side. **Your protocol should use both ACK and NACK messages.**

You should choose a very large value for the average time between messages from sender's layer5, so that your sender is never called while it still has an outstanding, unacknowledged message it is trying to send to the receiver. I'd suggest you choose a value of 1000. You should also perform a check in your sender to make sure that when A_output() is called, there is no message currently in transit. If there is, you can simply ignore (drop) the data being passed to the A_output() routine.

You should put your procedures in a file called prog2.c. You will need the initial version of this file, containing the emulation routines we have writen for you, and the stubs for your procedures. You can obtain this program from http://gaia.cs.umass.edu/kurose/transport/prog2.c.

This lab can be completed on any machine supporting C. It makes no use of UNIX features. (You can simply copy the prog2.c file to whatever machine and OS you choose).

We recommend that you should hand in a code listing, a screen shot of output, and an explanation of events in the output. For your sample output, your procedures might print out a message whenever an event occurs at your sender or receiver (a message/packet arrival, or a timer interrupt) as well as any action taken in response. You might want to hand in output for a run up to the point (approximately) when 10 messages have been ACK'ed correctly at the receiver, a loss probability of 0.1, and a corruption probability of 0.3, and a trace level of 2. You might want to annotate your printout showing how your protocol correctly recovered from packet loss and corruption.

Note 1: The code requires GCC 4.8.
Ubuntu 14.0.4 comes with GCC 4.8. So you may need to install Ubuntu 14.0.4 in a virtual machine.

Note 2: Some students have suggested to add the line " #include <stdlib.h>" and removing all instances of "char *malloc(). "

Student Questions

Lab 3 (Cont)

Helpful Hints and the like

Checksumming. You can use whatever approach for checksumming you want. Remember that the sequence number and ack field can also be corrupted. We would suggest a TCP-like checksum, which consists of the sum of the (integer) sequence and ack field values, added to a character-by-character sum of the payload field of the packet (i.e., treat each character as if it were an 8 bit integer and just add them together).

Note that any shared "state" among your routines needs to be in the form of global variables. Note also that any information that your procedures need to save from one invocation to the next must also be a global (or static) variable. For example, your routines will need to keep a copy of a packet for possible retransmission. It would probably be a good idea for such a data structure to be a global variable in your code. Note, however, that if one of your global variables is used by your sender side, that variable should **NOT** be accessed by the receiving side entity, since in real life, communicating entities connected only by a communication channel can not share global variables.

There is a float global variable called *time* that you can access from within your code to help you out with your diagnostics msgs.

START SIMPLE. Set the probabilities of loss and corruption to zero and test out your routines. Better yet, design and implement your procedures for the case of no loss and no corruption, and get them working first. Then handle the case of one of these probabilities being non-zero, and then finally both being non-zero.

Debugging. We'd recommend that you set the tracing level to 2 and put LOTS of printf's in your code while your debugging your procedures.

Random Numbers. The emulator generates packet loss and errors using a random number generator. Our past experience is that random number generators can vary widely from one machine to another. You may need to modify the random number generation code in the emulator we have suplied you. Our emulation routines have a test to see if the random number generator on your machine will work with our code. If you get an error message:

It is likely that random number generation on your machine is different from what this emulator expects. Please take a look at the routine jimsrand() in the emulator code. Sorry.

then you'll know you'll need to look at how random numbers are generated in the routine jimsrand(); see the comments in that routine. Continued on Slide 3-70…

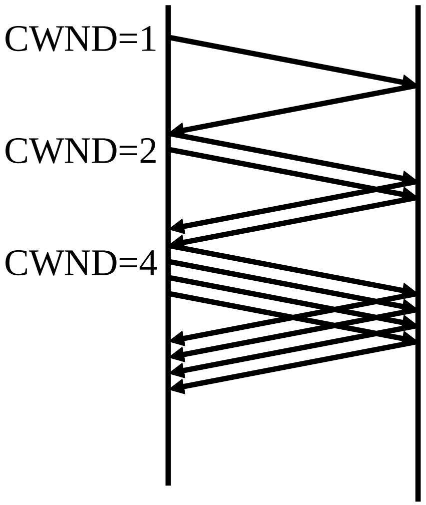
Student Questions

Optional Homework 3D

Try but do not submit.

A TCP entity opens a connection and uses slow start. Approximately how many round-trip times are required before TCP can send N segments. Assume no timeout.

Hint:



Student Questions

- ❑ Could you walk through the hint for problem 3D? *Sure.*
- ❑ Should we discuss it by the case for this question? (i.e., for large or low initial threshold value)

Not sure what the question is.

- ❑ Can I get extra credit if I submit it?

No. There is no place to submit it.

Acronyms

ACK	ACKnowledgement
AIMD	Additive increase and multiplicative decrease
ARQ	Automatic Repeat Request
CE	Congestion Experienced
CRC	Cyclic Redundancy Check
CWND	Congestion Window
CWR	Congestion Window Reduced
DA	Destination Address
DEC	Digital Equipment Corporation
DECbit	DEC's bit based congestion scheme
DevRTT	Deviation of RTT
DNS	Domain Name System
DP	Destination Port
ECE	Explicit Congestion Experienced
ECN	Explicit Congestion Notification
FIN	Final

Student Questions

Acronyms (Cont)

FTP	File Transfer Protocol
GBN	Go-Back N
HTTP	Hyper-Text Transfer Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISN	Initial Sequence Number
kB	Kilo-Byte
MSS	Maximum segment size
PBX	Private Branch Exchange
PSH	Push
RFC	Request for Comments
RM	Resource Management
RST	Reset
RTT	Round-Trip Time
SA	Source Address
SACK	Selective Acknolowledgement

Student Questions

Acronyms (Cont)

□ SMTP	Simple Mail Transfer Protocol
□ SP	Source Port
□ SSThresh	Slow Start Threshold
□ SYN	Synchronization
□ SYNACK	SYN Acknowledgement
□ TCP	Transmission Control Protocol
□ UDP	User Datagram Protocol
□ URG	Urgent
□ VCI	Virtual Circuit Identifiers

Student Questions

Scan This to Download These Slides

Raj Jain
http://rajjain.com

http://www.cse.wustl.edu/~jain/cse473-25/i_3tcp.htm

Student Questions

❑ Why neither TCP nor UDP provides bandwidth guarantees service?

The TCP/IP community was academic for a long time. Many attempts were made and failed. MPLS has succeeded.

❑ Domain names are unique across the Internet. What about hostnames? Some people say that in a local area network, hostnames are unique. Is this correct?

Yes. You cannot have two computers with the same name in one LAN. In the Internet, we use a "fully qualified domain name (FQDN)." No two computers should have the same FQDN.

❑ Why does the class start at least 5 mins late?

To achieve quorum and to set up all systems.

❑ What is the format of the exam?

See Sample Exam on Canvas.

Related Modules

CSE 567: The Art of Computer Systems Performance Analysis
https://www.youtube.com/playlist?list=PLjGG94etKypJEKjNAa1n_1X0bWWNyZcof

CSE473S: Introduction to Computer Networks (Fall 2011),
https://www.youtube.com/playlist?list=PLjGG94etKypJWOSPMh8Azcgy5e_10TiDw

CSE 570: Recent Advances in Networking (Spring 2013)
https://www.youtube.com/playlist?list=PLjGG94etKypLHyBN8mOgwJLHD2FFIMGq5

CSE571S: Network Security (Spring 2011),
https://www.youtube.com/playlist?list=PLjGG94etKypKvzfVtutHcPFJXumyyg93u

Video Podcasts of Prof. Raj Jain's Lectures,
https://www.youtube.com/channel/UCN4-5wzNP9-ruOzQMs-8NUw

Student Questions

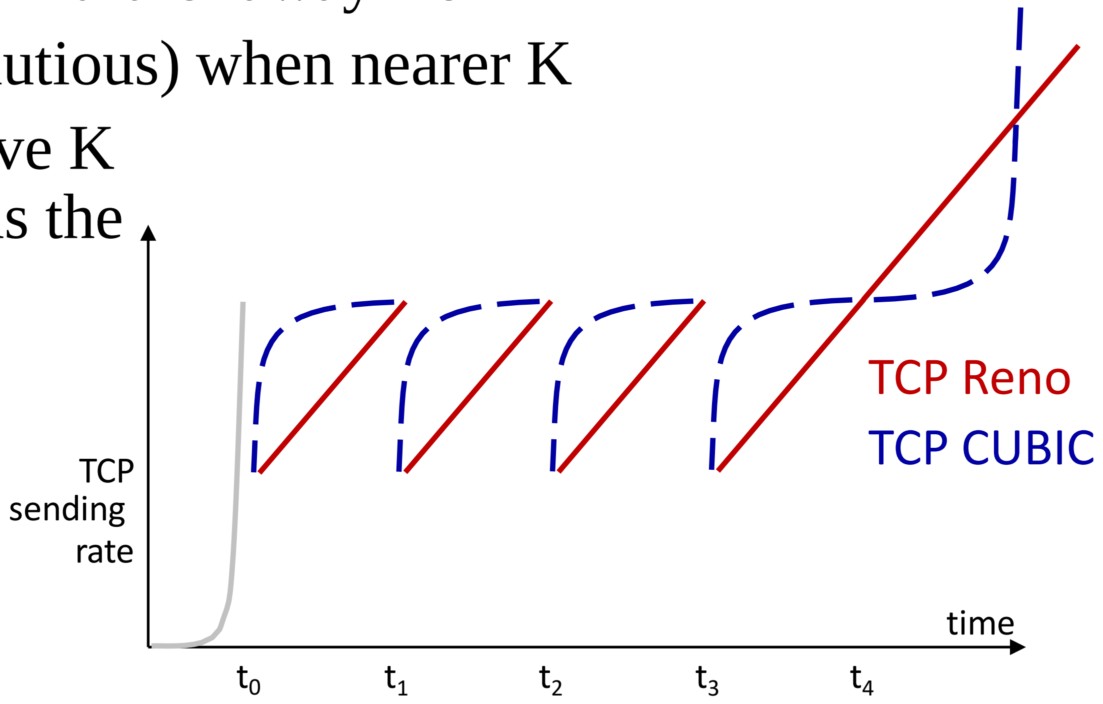
Network Utilities

- TCPview: Shows active ports on your system https://docs.microsoft.com/en-us/sysinternals/downloads/tcpview

Student Questions

TCP CUBIC

- K: Estimate of time when TCP window size will reach W_{max}. K is a tunable parameter.
- Increase W as a function of the *cube* of the distance between the current time and K
- Larger increases when further away from K
- Smaller increases (cautious) when nearer K
- Increases slowly above K and then quickly finds the new limit
- TCP QUBIC is default in Linux
- Most popular TCP for Web servers



Ref: L. Xu, S. Ha, A. Zimmermann, L. Eggert, R. Scheffenegger, "CUBIC for Fast Long Distance Networks," RFC 8312, Feb 2018.

Student Questions

- How is the estimate of time when TCP window size will reach W_max (K) calculated?

You can compute the time using linear prediction (red line).

- Can you discuss the difference between TCP Tahoe, Reno, and Cubic?

Yes. TCP Tahoe and Reno implemented Slowstart and its improvement. Cubit is a replacement for Slowstart. Cubic works well for long-distance or high-speed networks where window sizes are large.

Lab 3 Hints

- Some students received warning messages when trying to compile the C code. This can be fixed by adding the line

  #include <stdlib.h>

  and removing all instances of

  char *malloc();
- The method starttimer, which schedules timerinterrupt to trigger, is the one to be careful of.
  - starttimer schedules timerinterrupt to trigger after some amount of time and have the following signature.
  - starttimer(int calling_entity, float increment){}
    Note that the 2nd parameter must be a **float**.
  - If increment is not cast to a float, it leads to unexpected behavior (triggering back to back to back to back...).
  - This is an issue of bit representation. C will take in the bits that represent an int and interpret them as if they were a float. This leads to a passing in a much smaller value than expected.

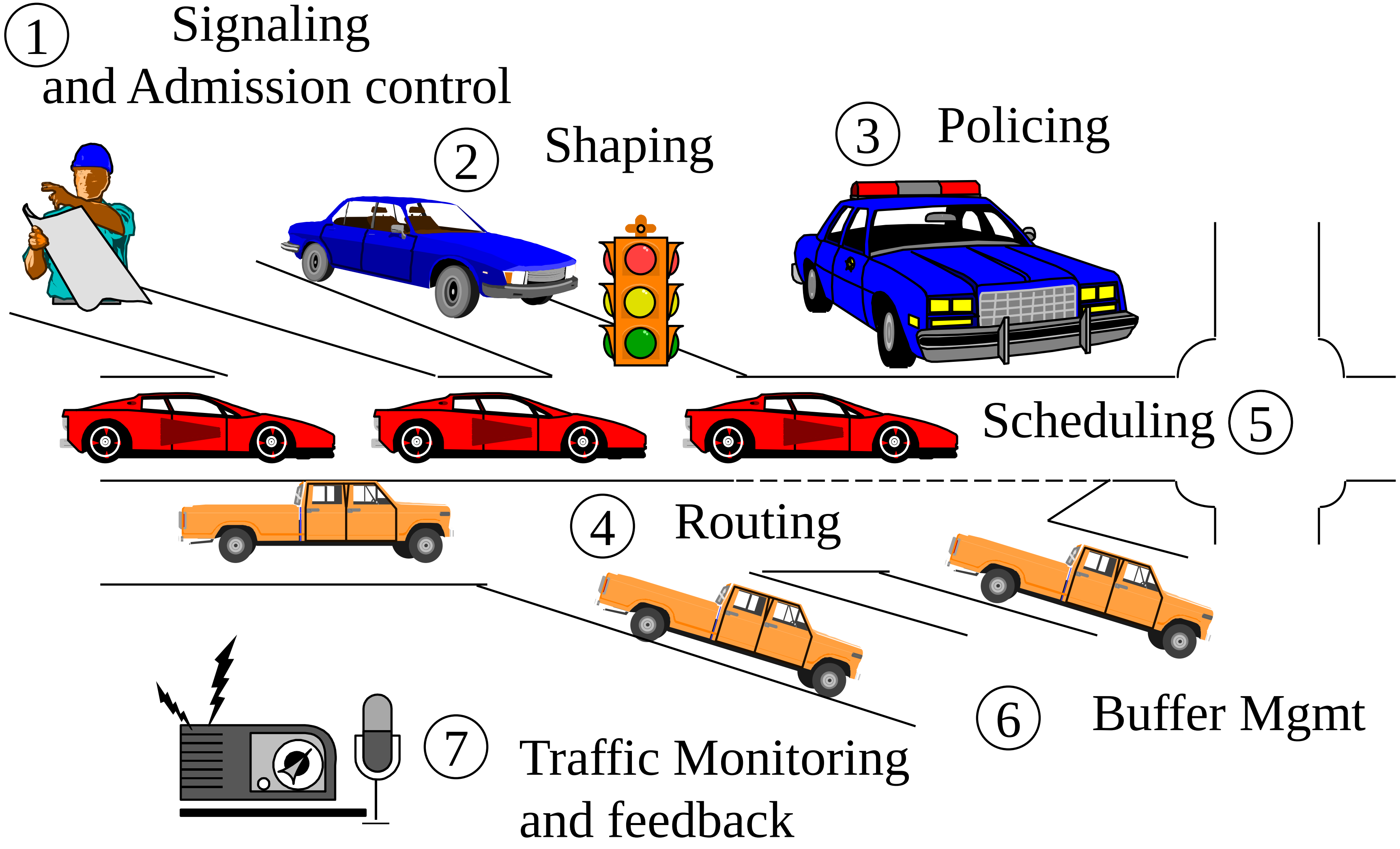
Student Questions

Lab 3 Hints (Cont)

- i.e., the following will trigger back to back to back...
- starttimer(A_is_calling_entity, 2000);
  // Leads to lots of repeated timerinterrupts
- The following will behave as expected
- starttimer(A_is_calling_entity, (float)2000); // Casting to float will fix the issue

Student Questions

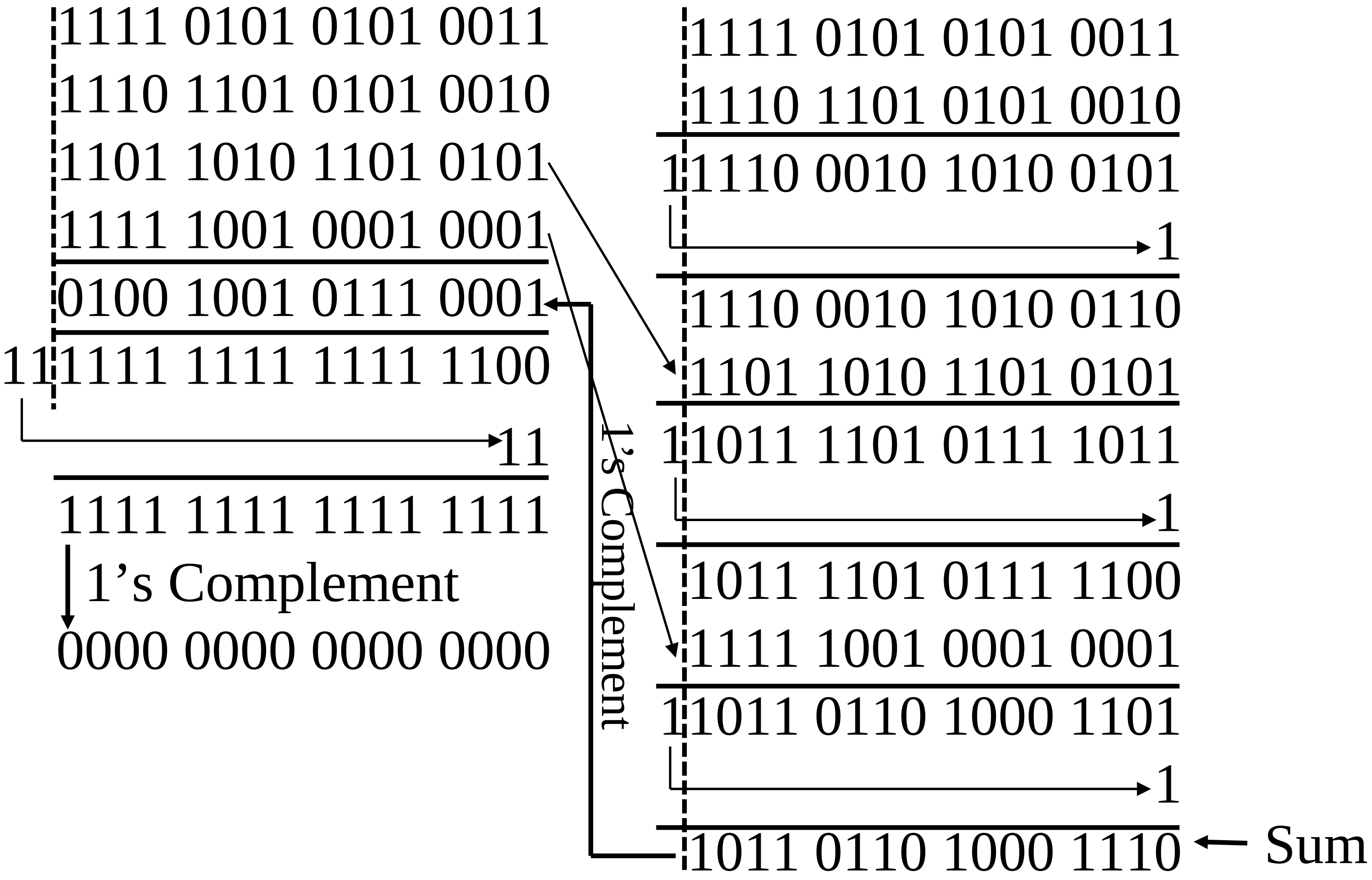
Traffic Management Methods



Student Questions

Checksum: Another Example

- Four 16-bit words:



1111 0101 0101 0011
1110 1101 0101 0010
1101 1010 1101 0101
1111 1001 0001 0001
0100 1001 0111 0001
11 1111 1111 1111 1100
11
1111 1111 1111 1111
1’s Complement
0000 0000 0000 0000

1’s Complement

1111 0101 0101 0011
1110 1101 0101 0010
1 1110 0010 1010 0101
1
1110 0010 1010 0110
1101 1010 1101 0101
1 1011 1101 0111 1011
1
1011 1101 0111 1100
1111 1001 0001 0001
1 1011 0110 1000 1101
1
1011 0110 1000 1110 ← Sum

Student Questions