

COS 140: Foundations of Computer Science

Transport-Layer Protocols*

Fall 2018

*This lecture draws heavily from Kurose & Ross (2008): *Computer Networks: A Top-Down Approach*

Homework

Transport Layer

Unreliable Transport
Protocols

Reliable Transport
Protocols

Example: TCP

Other Transport Layer
Topics

- Reading: None
- Slides online
- Homework:
 - On Blackboard
 - Due 12/10

Protocols

Transport Layer

- **Protocols**

- Transport layer
- Ports & sockets
- Types

Unreliable Transport Protocols

Reliable Transport Protocols

Example: TCP

Other Transport Layer Topics

- Recall: Protocol is a description of a pattern of interaction between agents
- Most common: TCP/IP

Review: Layered protocols

Transport Layer

● Protocols

- Transport layer
- Ports & sockets
- Types

Unreliable Transport Protocols

Reliable Transport Protocols

Example: TCP

Other Transport Layer Topics

- Recall: Layered *network stack*

Review: Layered protocols

Transport Layer

● Protocols

- Transport layer
- Ports & sockets
- Types

Unreliable Transport Protocols

Reliable Transport Protocols

Example: TCP

Other Transport Layer Topics

- Recall: Layered *network stack*
 - Application: network applications (e.g., Web clients/servers), application protocols (e.g., HTTP)

Review: Layered protocols

Transport Layer

● Protocols

- Transport layer
- Ports & sockets
- Types

Unreliable Transport Protocols

Reliable Transport Protocols

Example: TCP

Other Transport Layer Topics

- Recall: Layered *network stack*
 - Application: network applications (e.g., Web clients/servers), application protocols (e.g., HTTP)
 - Transport: Delivery of application messages between applications: error correction, reliable transport (some protocols), etc.

Review: Layered protocols

Transport Layer

● Protocols

- Transport layer
- Ports & sockets
- Types

Unreliable Transport Protocols

Reliable Transport Protocols

Example: TCP

Other Transport Layer Topics

- Recall: Layered *network stack*
 - Application: network applications (e.g., Web clients/servers), application protocols (e.g., HTTP)
 - Transport: Delivery of application messages between applications: error correction, reliable transport (some protocols), etc.
 - Network: Delivery of transport-layer messages: routing, etc.

Review: Layered protocols

Transport Layer

● Protocols

- Transport layer
- Ports & sockets
- Types

Unreliable Transport Protocols

Reliable Transport Protocols

Example: TCP

Other Transport Layer Topics

- Recall: Layered *network stack*
 - Application: network applications (e.g., Web clients/servers), application protocols (e.g., HTTP)
 - Transport: Delivery of application messages between applications: error correction, reliable transport (some protocols), etc.
 - Network: Delivery of transport-layer messages: routing, etc.
 - Data link: Delivery of network-layer messages – e.g., forwarding to next router/host (e.g., Ethernet, WiFi, link layers on routers)

Review: Layered protocols

Transport Layer

● Protocols

- Transport layer
- Ports & sockets
- Types

Unreliable Transport Protocols

Reliable Transport Protocols

Example: TCP

Other Transport Layer Topics

- Recall: Layered *network stack*
 - Application: network applications (e.g., Web clients/servers), application protocols (e.g., HTTP)
 - Transport: Delivery of application messages between applications: error correction, reliable transport (some protocols), etc.
 - Network: Delivery of transport-layer messages: routing, etc.
 - Data link: Delivery of network-layer messages – e.g., forwarding to next router/host (e.g., Ethernet, WiFi, link layers on routers)
 - Physical: Moving the bits (Ethernet has variants for different media)

Layered protocols

Transport Layer

● **Protocols**

- Transport layer
- Ports & sockets
- Types

Unreliable Transport Protocols

Reliable Transport Protocols

Example: TCP

Other Transport Layer Topics

- As message goes from top → bottom:
 - Broken into pieces
 - Each piece has its own header added at each level

Transport layer

Transport Layer

- Protocols
- **Transport layer**
- Ports & sockets
- Types

Unreliable Transport Protocols

Reliable Transport Protocols

Example: TCP

Other Transport Layer Topics

- Responsible for delivery of application-layer messages between processes
- Interacts with: application and network layers
- Divides/encapsulates application messages as *segments*
- Possibly:
 - Error correction
 - Reliable delivery
 - Congestion control
- Connectionless (e.g., UDP) versus connection-oriented (e.g., TCP)

Ports and sockets

Transport Layer

- Protocols
- Transport layer
- Ports & sockets
- Types

Unreliable Transport Protocols

Reliable Transport Protocols

Example: TCP

Other Transport Layer Topics

- Network stack manages set of *ports* on host:
 - Interface of host to outside world
 - Numbered (virtual) connection points
 - Some: well-known ports (e.g., email (25), Web (80))
 - Some: usable on fly by processes
- *Sockets*: virtual connections between application process and transport layer (and thus, a port)
- Each application process talks to (e.g.) TCP via a socket
- Can have multiple sockets attached to a port
- E.g.: Web server
- Transport layer: multiplexing/demultiplexing of sockets ↔ ports

Ports and sockets

Transport Layer

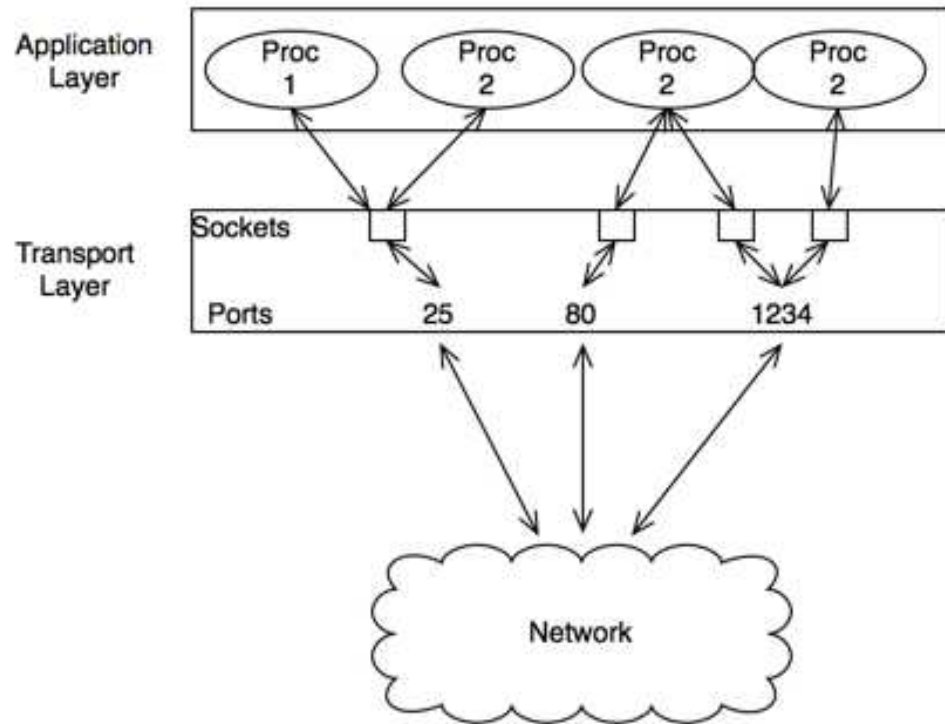
- Protocols
- Transport layer
- Ports & sockets
- Types

Unreliable Transport Protocols

Reliable Transport Protocols

Example: TCP

Other Transport Layer Topics



Connection-oriented and connectionless protocols

Transport Layer

- Protocols
- Transport layer
- Ports & sockets
- **Types**

Unreliable Transport Protocols

Reliable Transport Protocols

Example: TCP

Other Transport Layer Topics

- Connectionless protocols:
 - E.g., User Datagram Protocol (UDP)
 - Send segments (“datagrams”) between application layers
 - No notion of a continuous connection – think US Mail
 - Maybe some error checking embedded
 - No error correcting
 - Not reliable

Connection-oriented and connectionless protocols

Transport Layer

- Protocols
- Transport layer
- Ports & sockets
- **Types**

Unreliable Transport Protocols

Reliable Transport Protocols

Example: TCP

Other Transport Layer Topics

- Connectionless protocols:
 - E.g., User Datagram Protocol (UDP)
 - Send segments (“datagrams”) between application layers
 - No notion of a continuous connection – think US Mail
 - Maybe some error checking embedded
 - No error correcting
 - Not reliable
- Connection-oriented protocols:
 - E.g., Transmission Control Protocol (TCP)
 - Connection: a virtual pipeline between applications
 - Think “phone”, but no real connection
 - Reliable transport protocols

Unreliable Transport Protocols

Transport Layer

Unreliable Transport
Protocols

Reliable Transport
Protocols

Example: TCP

Other Transport Layer
Topics

- Very simple
- Possibly segment application data
- Possibly add error-checking code (e.g., CRC)
- Just pass along to network layer

Reliable transport protocols

Transport Layer

Unreliable Transport
Protocols

Reliable Transport
Protocols

- Using a reliable channel
- Bit errors
- Bit error protocol
- Lossy channels
- Lossy protocol
- Pipelining
- Recovery in pipelining

Example: TCP

Other Transport Layer
Topics

- More complex
- Have to deal with:
 - bitwise errors
 - lost segments
 - out-of-order segments
- Can conceptualize as *state machines*
- We'll look at increasingly-complex variants

Using a reliable channel

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

- Using a reliable channel

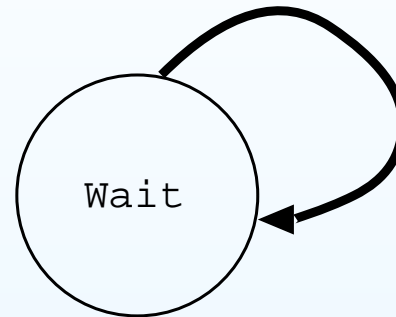
- Bit errors
- Bit error protocol
- Lossy channels
- Lossy protocol
- Pipelining
- Recovery in pipelining

Example: TCP

Other Transport Layer Topics

- Simplest

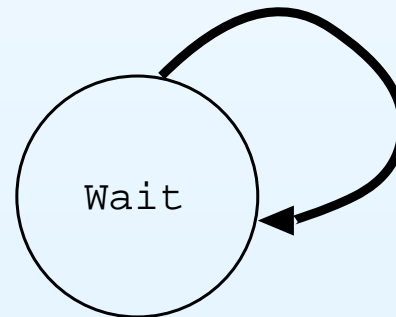
Sender :



Request from application layer

Create packet (segment)
Send packet

Receiver :



Receive packet

Extract data
Deliver to application

Using a channel with bit errors

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

- Using a reliable channel
- **Bit errors**
- Bit error protocol
- Lossy channels
- Lossy protocol
- Pipelining
- Recovery in pipelining

Example: TCP

Other Transport Layer Topics

- Problem: Few channels are reliable
- Simple problem: errors in some bits
- Detect this with: parity, cyclic redundancy checks (CRC)
- The question is: what to do when detected?

Using a channel with bit errors

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

- Using a reliable channel
- Bit errors
- Bit error protocol
- Lossy channels
- Lossy protocol
- Pipelining
- Recovery in pipelining

Example: TCP

Other Transport Layer Topics

- For this, need additional messages:
 - ACK: to acknowledge correct receipt
 - NAK: negative acknowledgment
- Recipient checks the packet, sends the appropriate message in reply
- Problem: ACKs and NAKs can be garbled, too!

A protocol for handling bit errors

Transport Layer

Unreliable Transport Protocols

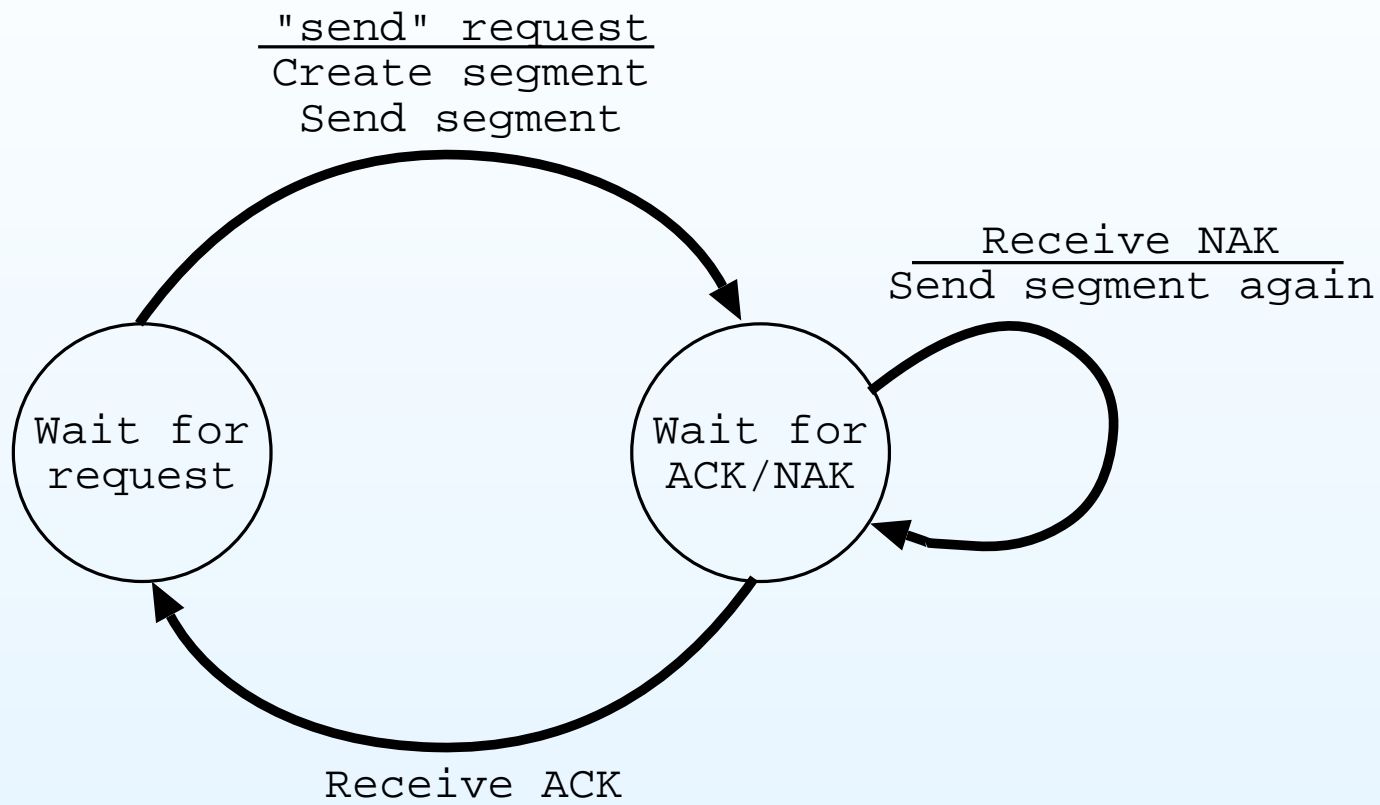
Reliable Transport Protocols

- Using a reliable channel
- Bit errors
- **Bit error protocol**
- Lossy channels
- Lossy protocol
- Pipelining
- Recovery in pipelining

Example: TCP

Other Transport Layer Topics

- **Sender:**



A protocol for handling bit errors

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

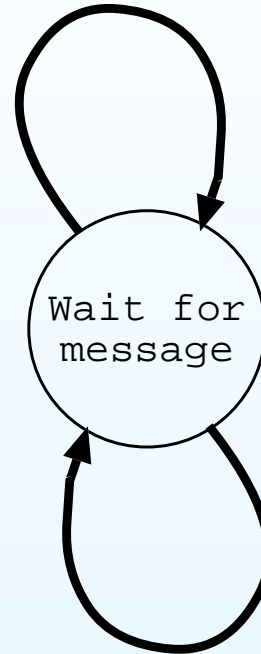
- Using a reliable channel
- Bit errors
- **Bit error protocol**
- Lossy channels
- Lossy protocol
- Pipelining
- Recovery in pipelining

Example: TCP

Other Transport Layer Topics

- Recipient:

Corrupt segment
Create NAK segment
Send segment



Good segment
Extract data
Deliver data
Create ACK segment
Send segment

A protocol for handling bit errors

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

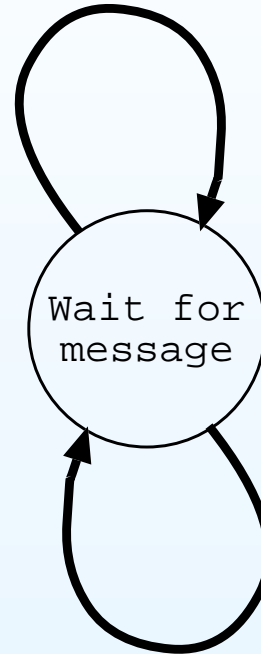
- Using a reliable channel
- Bit errors
- **Bit error protocol**
- Lossy channels
- Lossy protocol
- Pipelining
- Recovery in pipelining

Example: TCP

Other Transport Layer Topics

- Recipient:

Corrupt segment
Create NAK segment
Send segment



Good segment
Extract data
Deliver data
Create ACK segment
Send segment

- Any problems with this?

A protocol for handling bit errors

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

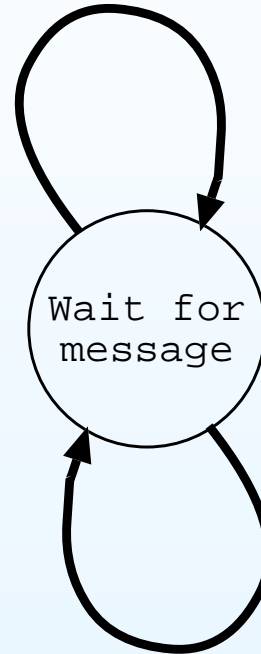
- Using a reliable channel
- Bit errors
- **Bit error protocol**
- Lossy channels
- Lossy protocol
- Pipelining
- Recovery in pipelining

Example: TCP

Other Transport Layer Topics

- Recipient:

Corrupt segment
Create NAK segment
Send segment



Good segment
Extract data
Deliver data
Create ACK segment
Send segment

- Any problems with this? Yes! What about errors in ACK/NAK?

A better protocol for handling bit errors

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

- Using a reliable channel
- Bit errors
- **Bit error protocol**
- Lossy channels
- Lossy protocol
- Pipelining
- Recovery in pipelining

Example: TCP

Other Transport Layer Topics

- Have the sender number its segments
- Receiver can then determine if the packet received is a retransmission
- ACK/NAK don't need to say what they're ACK'ing (or not) – since no messages are lost, garbled or okay, the last ACK/NAK was for its last message
- For this simple protocol, we only need two sequence numbers, 0 and 1 (a bit) – only one packet being dealt with at a time.

A better protocol for handling bit errors

Transport Layer

Unreliable Transport Protocols

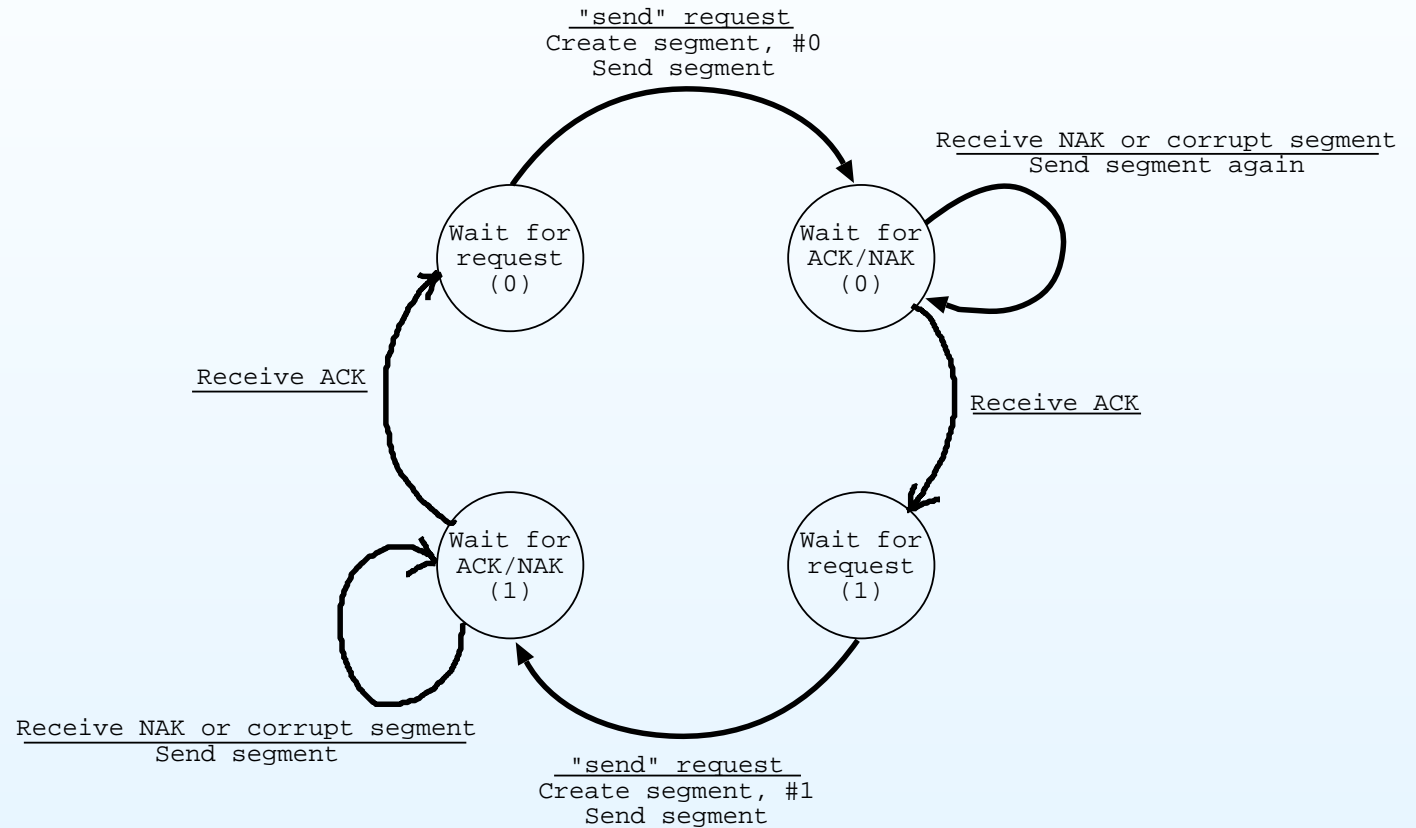
Reliable Transport Protocols

- Using a reliable channel
- Bit errors
- **Bit error protocol**
- Lossy channels
- Lossy protocol
- Pipelining
- Recovery in pipelining

Example: TCP

Other Transport Layer Topics

- **Sender:**



A better protocol for handling bit errors

Transport Layer

Unreliable Transport Protocols

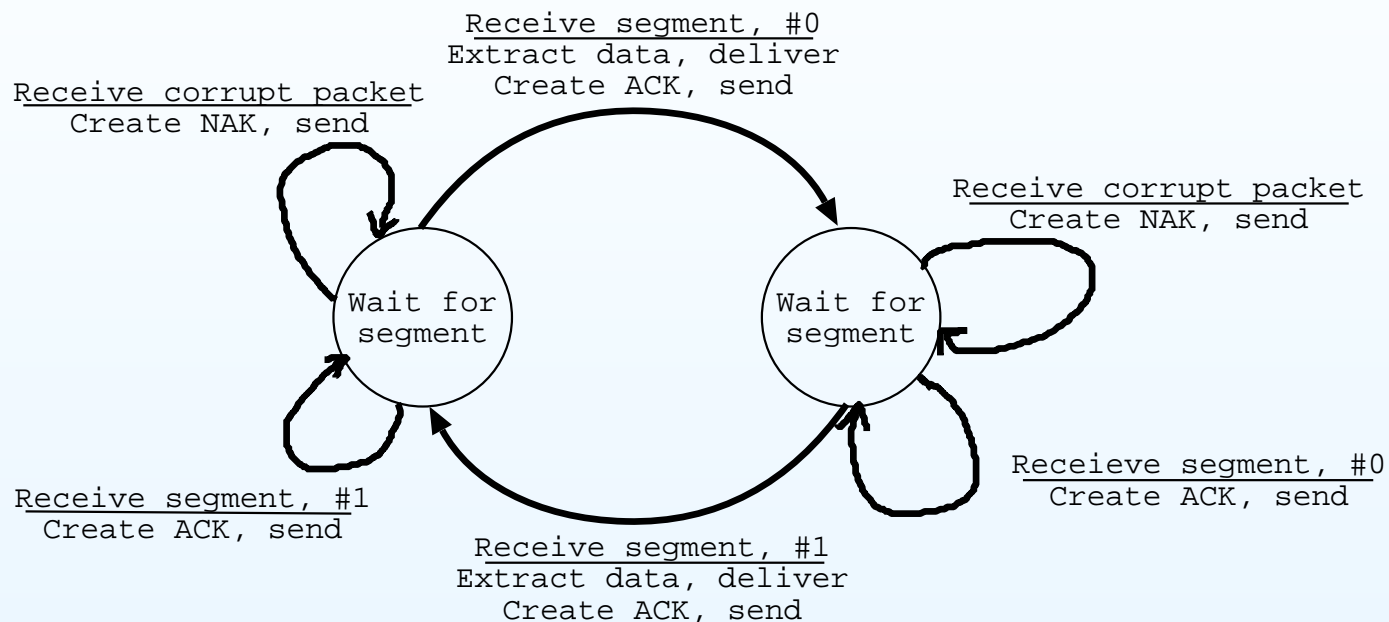
Reliable Transport Protocols

- Using a reliable channel
- Bit errors
- **Bit error protocol**
- Lossy channels
- Lossy protocol
- Pipelining
- Recovery in pipelining

Example: TCP

Other Transport Layer Topics

- Recipient:



Using a “lossy” channel

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

- Using a reliable channel
- Bit errors
- Bit error protocol
- **Lossy channels**
- Lossy protocol
- Pipelining
- Recovery in pipelining

Example: TCP

Other Transport Layer Topics

- Even harder: What if the channel can lose some messages?
- ACKs and NAKs could also be lost or garbled.
- Need timers, now – if haven’t received an ACK after some time, retransmit
- How to choose the time?

Using a “lossy” channel

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

- Using a reliable channel
- Bit errors
- Bit error protocol
- **Lossy channels**
- Lossy protocol
- Pipelining
- Recovery in pipelining

Example: TCP

Other Transport Layer Topics

- Even harder: What if the channel can lose some messages?
- ACKs and NAKs could also be lost or garbled.
- Need timers, now – if haven’t received an ACK after some time, retransmit
- How to choose the time? At least round trip delay + some

Protocol for lossy channels

Transport Layer

Unreliable Transport Protocols

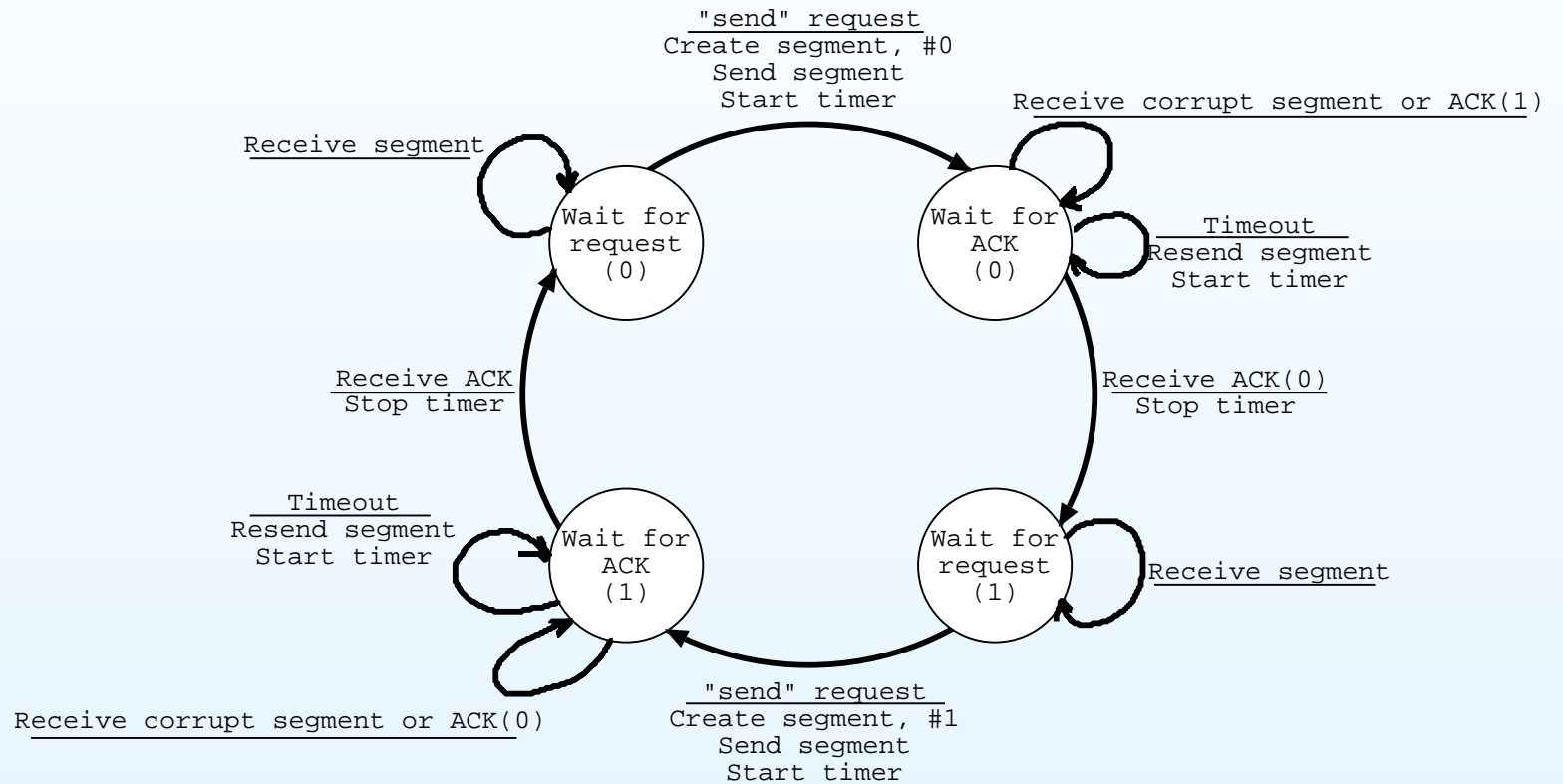
Reliable Transport Protocols

- Using a reliable channel
- Bit errors
- Bit error protocol
- Lossy channels
- **Lossy protocol**
- Pipelining
- Recovery in pipelining

Example: TCP

Other Transport Layer Topics

- **Sender:**



Pipelining

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

- Using a reliable channel
- Bit errors
- Bit error protocol
- Lossy channels
- Lossy protocol
- **Pipelining**
- Recovery in pipelining

Example: TCP

Other Transport Layer Topics

- Problem: reliable, but inefficient
- Suppose it takes 5 ms for message to propagate from sender to receiver, or vice versa – a 1 Gbps channel, and time to actually put the message on the channel is negligible
- Time to send, say, a 1 KB message:
 - Send message: 5 ms
 - Send ACK: 5 ms
 - Total 10 ms/message
 - Transfer rate = 1 KB/10 ms = only 100 KB/s!

Pipelining

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

- Using a reliable channel
- Bit errors
- Bit error protocol
- Lossy channels
- Lossy protocol
- **Pipelining**
- Recovery in pipelining

Example: TCP

Other Transport Layer Topics

- Better idea: don't wait for ACK before sending other messages
- Now can have n messages “in the pipe” at once.
- How many, potentially?

Pipelining

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

- Using a reliable channel
- Bit errors
- Bit error protocol
- Lossy channels
- Lossy protocol
- **Pipelining**
- Recovery in pipelining

Example: TCP

Other Transport Layer Topics

- Better idea: don't wait for ACK before sending other messages
- Now can have n messages “in the pipe” at once.
- How many, potentially?
 - If 1 Gbps channel, 1 KB messages, then a message takes

$$\frac{8Kb}{1Gb/s} = \frac{2^{13}}{2^{30}}s = 2^{-17}s \approx 7.6\mu s$$

Pipelining

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

- Using a reliable channel
- Bit errors
- Bit error protocol
- Lossy channels
- Lossy protocol
- **Pipelining**
- Recovery in pipelining

Example: TCP

Other Transport Layer Topics

- Better idea: don't wait for ACK before sending other messages
- Now can have n messages “in the pipe” at once.
- How many, potentially?
 - If 1 Gbps channel, 1 KB messages, then a message takes

$$\frac{8Kb}{1Gb/s} = \frac{2^{13}}{2^{30}}s = 2^{-17}s \approx 7.6\mu s$$

- So there can be $\frac{5ms}{7.6\mu s} \approx 658$ messages in the pipe at once
- In practice, have (far) fewer:
 - *Window*: what can be sent before an ACK received
 - Receive an ACK: slide window, can transmit more

Recovery in pipelining

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

- Using a reliable channel
- Bit errors
- Bit error protocol
- Lossy channels
- Lossy protocol
- Pipelining
- **Recovery in pipelining**

Example: TCP

Other Transport Layer Topics

- In send-and-wait protocols, pretty clear what you're ACKing
- What about pipelined protocols?
- Have to mark ACK with what is being acknowledged
- What to do when one is missing/corrupt?

Recovery in pipelining

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

- Using a reliable channel
- Bit errors
- Bit error protocol
- Lossy channels
- Lossy protocol
- Pipelining
- **Recovery in pipelining**

Example: TCP

Other Transport Layer Topics

- In send-and-wait protocols, pretty clear what you're ACKing
- What about pipelined protocols?
- Have to mark ACK with what is being acknowledged
- What to do when one is missing/corrupt?
 - Go-Back-N: When missing one (corrupt or timeout), repeat it and all others after it

Recovery in pipelining

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

- Using a reliable channel
- Bit errors
- Bit error protocol
- Lossy channels
- Lossy protocol
- Pipelining
- **Recovery in pipelining**

Example: TCP

Other Transport Layer Topics

- In send-and-wait protocols, pretty clear what you're ACKing
- What about pipelined protocols?
- Have to mark ACK with what is being acknowledged
- What to do when one is missing/corrupt?
 - Go-Back-N: When missing one (corrupt or timeout), repeat it and all others after it
 - Selective repeat: Just repeat the one missing

Recovery in pipelining

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

- Using a reliable channel
- Bit errors
- Bit error protocol
- Lossy channels
- Lossy protocol
- Pipelining
- **Recovery in pipelining**

Example: TCP

Other Transport Layer Topics

- In send-and-wait protocols, pretty clear what you're ACKing
- What about pipelined protocols?
- Have to mark ACK with what is being acknowledged
- What to do when one is missing/corrupt?
 - Go-Back-N: When missing one (corrupt or timeout), repeat it and all others after it
 - Selective repeat: Just repeat the one missing
- GBN simpler, not as efficient

Example: TCP

Transport Layer

Unreliable Transport
Protocols

Reliable Transport
Protocols

Example: TCP

- Segments
- Sequence numbers
- Connection initiation
- Handling problems
- Handling problems

Other Transport Layer
Topics

- Connection-oriented, reliable data transport protocol
- Can handle bit errors, lossy channels
- Full-duplex

TCP segment structure

Transport Layer

Unreliable Transport
Protocols

Reliable Transport
Protocols

Example: TCP

- Segments

- Sequence numbers
- Connection initiation
- Handling problems
- Handling problems

Other Transport Layer
Topics

- Header + data
- Source port number, destination port number
- Sequence number
- ACK number
- Checksum
- Header length field, flags, options, some other stuff

Sequence numbers

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

Example: TCP

- Segments
- **Sequence numbers**
- Connection initiation
- Handling problems
- Handling problems

Other Transport Layer Topics

- Each half of the conversation is considered an ordered sequence of bytes
- Sequence number of a segment is the byte number of the first byte in the segment – not the segment number!
- ACK number: The next byte expected from the sender
- These are *cumulative acknowledgments*

Connection initiation: “three-way handshake”

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

Example: TCP

- Segments
- Sequence numbers
- **Connection initiation**
- Handling problems
- Handling problems

Other Transport Layer Topics

- First: Client sends a special segment (“SYN segment”) to request connection
 - No application data contained
 - SYN bit in header = 1
 - Random sequence number
- Second: Server sets up its side of the connection and sends message 2 (“SYNACK segment”)
 - Allocates buffers, variables
 - Response segment: no application data
 - SYN = 1, ACK = client sequence number + 1, random sequence number

Connection initiation: “three-way handshake”

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

Example: TCP

- Segments
- Sequence numbers
- **Connection initiation**
- Handling problems
- Handling problems

Other Transport Layer Topics

- Third: Client sets up its side, sends another message
 - Allocates client-side buffers, variables
 - Segment has SYN = 0, server's sequence number +1 as ACK
 - Can carry application data (*payload*)

Handling problems

Transport Layer

Unreliable Transport
Protocols

Reliable Transport
Protocols

Example: TCP

- Segments
- Sequence numbers
- Connection initiation
- **Handling problems**
- Handling problems

Other Transport Layer
Topics

- Corrupt segment received (bitwise error) –

Handling problems

Transport Layer

Unreliable Transport
Protocols

Reliable Transport
Protocols

Example: TCP

- Segments
- Sequence numbers
- Connection initiation
- **Handling problems**
- Handling problems

Other Transport Layer
Topics

- Corrupt segment received (bitwise error) – don't ACK

Handling problems

Transport Layer

Unreliable Transport
Protocols

Reliable Transport
Protocols

Example: TCP

- Segments
- Sequence numbers
- Connection initiation
- **Handling problems**
- Handling problems

Other Transport Layer
Topics

- Corrupt segment received (bitwise error) – don't ACK
- Receive duplicate segment –

Handling problems

Transport Layer

Unreliable Transport
Protocols

Reliable Transport
Protocols

Example: TCP

- Segments
- Sequence numbers
- Connection initiation
- **Handling problems**
- Handling problems

Other Transport Layer
Topics

- Corrupt segment received (bitwise error) – don't ACK
- Receive duplicate segment –
 - Why? Lost ACK

Handling problems

Transport Layer

Unreliable Transport
Protocols

Reliable Transport
Protocols

Example: TCP

- Segments
- Sequence numbers
- Connection initiation
- **Handling problems**
- Handling problems

Other Transport Layer
Topics

- Corrupt segment received (bitwise error) – don't ACK
- Receive duplicate segment –
 - Why? Lost ACK
 - Just discard data, re-ACK

Handling problems

Transport Layer

Unreliable Transport
Protocols

Reliable Transport
Protocols

Example: TCP

- Segments
- Sequence numbers
- Connection initiation
- **Handling problems**
- Handling problems

Other Transport Layer
Topics

- Corrupt segment received (bitwise error) – don't ACK
- Receive duplicate segment –
 - Why? Lost ACK
 - Just discard data, re-ACK
- Timeout –

Handling problems

Transport Layer

Unreliable Transport
Protocols

Reliable Transport
Protocols

Example: TCP

- Segments
- Sequence numbers
- Connection initiation
- **Handling problems**
- Handling problems

Other Transport Layer
Topics

- Corrupt segment received (bitwise error) – don't ACK
- Receive duplicate segment –
 - Why? Lost ACK
 - Just discard data, re-ACK
- Timeout –
 - Why? Lost segment

Handling problems

Transport Layer

Unreliable Transport
Protocols

Reliable Transport
Protocols

Example: TCP

- Segments
- Sequence numbers
- Connection initiation
- **Handling problems**
- Handling problems

Other Transport Layer
Topics

- Corrupt segment received (bitwise error) – don't ACK
- Receive duplicate segment –
 - Why? Lost ACK
 - Just discard data, re-ACK
- Timeout –
 - Why? Lost segment
 - Single timer for all messages to reduce overhead

Handling problems

Transport Layer

Unreliable Transport
Protocols

Reliable Transport
Protocols

Example: TCP

- Segments
- Sequence numbers
- Connection initiation
- **Handling problems**
- Handling problems

Other Transport Layer
Topics

- Corrupt segment received (bitwise error) – don't ACK
- Receive duplicate segment –
 - Why? Lost ACK
 - Just discard data, re-ACK
- Timeout –
 - Why? Lost segment
 - Single timer for all messages to reduce overhead
 - Retransmit segment

Handling problems

Transport Layer

Unreliable Transport
Protocols

Reliable Transport
Protocols

Example: TCP

- Segments
- Sequence numbers
- Connection initiation
- Handling problems
- **Handling problems**

Other Transport Layer
Topics

- ACK received for segment *after* one it's expecting an ACK for –

Handling problems

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

Example: TCP

- Segments
- Sequence numbers
- Connection initiation
- Handling problems
- **Handling problems**

Other Transport Layer Topics

- ACK received for segment *after* one it's expecting an ACK for –
 - Not really a problem
 - Cumulative acknowledgment, so previous segments received, too

Handling problems

Transport Layer

Unreliable Transport
Protocols

Reliable Transport
Protocols

Example: TCP

- Segments
- Sequence numbers
- Connection initiation
- Handling problems
- **Handling problems**

Other Transport Layer
Topics

- Segment received out of order –

Handling problems

Transport Layer

Unreliable Transport
Protocols

Reliable Transport
Protocols

Example: TCP

- Segments
- Sequence numbers
- Connection initiation
- Handling problems
- **Handling problems**

Other Transport Layer
Topics

- Segment received out of order –
 - One reason: a segment was lost

Handling problems

Transport Layer

Unreliable Transport
Protocols

Reliable Transport
Protocols

Example: TCP

- Segments
- Sequence numbers
- Connection initiation
- Handling problems
- **Handling problems**

Other Transport Layer
Topics

- Segment received out of order –
 - One reason: a segment was lost
 - For this, send ACK, but with next byte expected being the missing segment

Handling problems

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

Example: TCP

- Segments
- Sequence numbers
- Connection initiation
- Handling problems
- **Handling problems**

Other Transport Layer Topics

- Segment received out of order –
 - One reason: a segment was lost
 - For this, send ACK, but with next byte expected being the missing segment
 - Another reason: segment tied up in network

Handling problems

Transport Layer

Unreliable Transport Protocols

Reliable Transport Protocols

Example: TCP

- Segments
- Sequence numbers
- Connection initiation
- Handling problems
- **Handling problems**

Other Transport Layer Topics

- Segment received out of order –
 - One reason: a segment was lost
 - For this, send ACK, but with next byte expected being the missing segment
 - Another reason: segment tied up in network
 - For this, ACK with next real byte expected, necessarily one right after this one

Other transport layer topics

Transport Layer

Unreliable Transport
Protocols

Reliable Transport
Protocols

Example: TCP

Other Transport Layer
Topics

- How to choose timeouts
- Flow control
- Congestion control
- When to use UDP vs TCP