

COMS3200

COMS3200 – Week 3 End-to-end Communication

School of Information Technology and Electrical Engineering
The University of Queensland

COMS3200

Outline

- IP and layers below – brief overview
- Reliable delivery of messages
- UDP
- TCP
 - Header
 - Connection establishment and teardown
- Flow control
 - Sliding Window Protocol
- Congestion control
- Assignment one discussion } Friday

2

COMS3200

Context ...

- Last week
 - Message passing
 - RPC
 - Data representation
- Next week
 - Physical layer

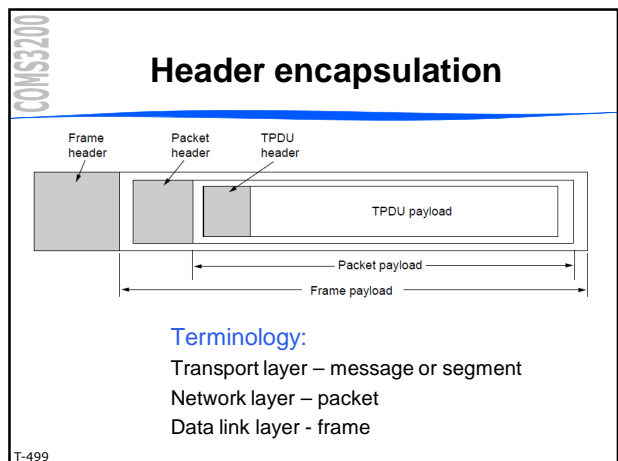
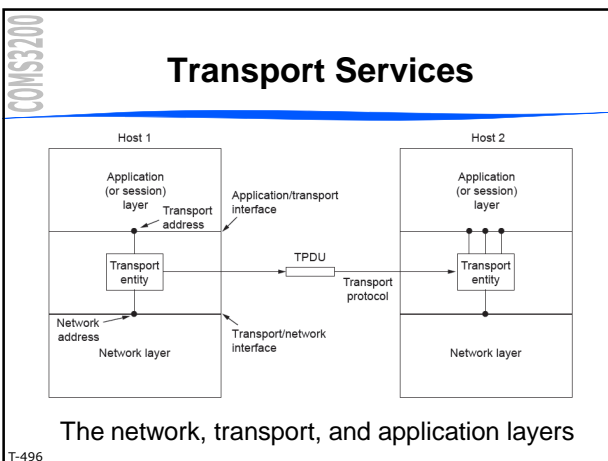
3

COMS3200

This Week's Learning Objectives

- Know the basic characteristics of UDP and TCP and when each would be used
- Understand the purpose of the various header fields for TCP and UDP
- Understand how TCP connections are established and torn down
- Understand flow control (sliding window) and congestion control in TCP

4



COMS3200

End-to-end Options

- Two transport options
 - Unreliable datagrams (connectionless)
 - UDP = User Datagram Protocol
 - Reliable connections
 - TCP = Transmission Control Protocol

7

COMS3200

IP layer and below

- IP layer routes packets to destination
 - is not reliable
 - packets may be dropped by routers
 - packets may be sent in wrong direction (and therefore delayed)
- Data Link layer delivers to adjacent node
 - may not be reliable (discards erroneous frames but may not tell sender)
 - limits frame size
- Physical Layer may garble frames

8

COMS3200

What do we have ?

- Best-effort packet delivery
- Packets may be
 - dropped
 - reordered
 - duplicated
 - size-limited
 - delayed

9

COMS3200

What do we want?

- Messages
 - transported from one host to another
 - guaranteed delivery (usually)
 - delivered in same order as sent
 - arbitrary size

10

COMS3200

What do we want? (cont.)

11

COMS3200

End-to-end Options

- Two transport options
 - Unreliable datagrams (connectionless)
 - UDP = User Datagram Protocol
 - Reliable connections
 - TCP = Transmission Control Protocol

12

COMS3200

Addressing in TCP and UDP

- Each protocol layer has its own addressing
- Transport addresses are **ports**
 - IP address uniquely addresses computing device
 - Port uniquely identifies application on this device

➔ socket = IP address + port

T-509, PD-371 to 372 13

COMS3200

UDP: User Datagram Protocol

- Connectionless Transport Protocol
- Allows applications to send datagrams to other applications
 - Basically an interface to IP
- No need to establish connection
- 8 byte header + data
- Header:

32 Bits	
Source port	Destination port
UDP length	UDP checksum

T-541 to 543, PD-370, RFC-768 14

COMS3200

Why would we want to use UDP?

- Don't care if packet lost
 - e.g. Streaming
- Small messages & reliable network
 - No connection establishment overhead
- No flow control
 - Can send things as fast as possible
- Simple implementation
 - No connection state information

15

COMS3200

UDP: Header

32 Bits	
Source port	Destination port
UDP length	UDP checksum

- Ports
 - Identify end-points within machines
 - e.g. processes
 - Source port is optional
- UDP Length
 - Includes header
- UDP Checksum
 - Includes header, data, pseudo-header
 - Optional – store 0

T-525 to 526 16

COMS3200

Reliable delivery

- Reliable delivery protocols use
 - acknowledgements
 - timeouts

17

COMS3200

Reliable delivery

- What happens if messages or ACKs are lost?
- Solution:
 - Setup **timer**
 - When timer is off, **retransmit** the message

18

Acknowledgements & Timeouts

(a) (b) (c) (d)

19

TCP: Transmission Control Protocol

- Connection-oriented
- Reliable (over unreliable internet network)
- Byte-stream
 - Not message stream
- Full-duplex
- Point-to-point (end-to-end)
 - Not multicast or broadcast
- Each machine has a *TCP transport entity*
 - user process or part of kernel

T-552 to 553, PD-371 to 372

20

TCP Responsibilities

- Sender
 - Accepts data streams from local processes
 - Breaks data into pieces < 64k bytes
 - Sends each piece as IP datagram
 - Time-out and retransmit as necessary
- Receiver
 - IP datagrams containing TCP data are passed to TCP entity for reconstruction
 - Acknowledge receipt
 - Reassemble datagrams in proper sequence

21

TCP Service Model: Sockets

- Sender and Receiver create sockets
- Sockets have addresses (IDs)
 - IP address + **port number**
- Multiple connections possible through one socket
 - Connections identified by socket ID of each end

22

Port Numbers

- 16 bits: 0 – 65535
- Below 1024
 - Well known ports
 - Reserved for standard services, e.g.
 - 23 - Telnet
 - 21 - FTP
 - 80 - HTTP
 - Look in /etc/services on a UNIX box

T-554

23

TCP: Byte Stream

- Message boundaries are not preserved
- Example
 - Sending process writes four 512 byte chunks
 - Receiver may get
 - one 2048 byte chunk (see below)
 - two 1024 byte chunks
 - many other possibilities
 - TCP doesn't care what the bytes are

(a) (b)

T-555, PD-375 to 376

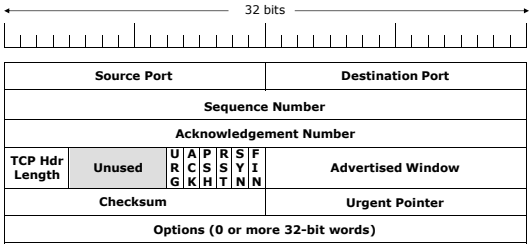
24

TCP Protocol

- TCP entities exchange **segments**
 - (another name for TPDU)
 - 20 byte header plus data
- Size limited by
 - IP packet size (max 65535 bytes)
 - Network – maximum transfer unit (MTU)
 - MTU = Max frame data (i.e. IP packet)
 - e.g. 1500 bytes for Ethernet
- Segments with zero data are valid

T-556 25

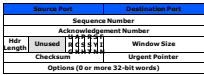
TCP Segment Header



T-557 26

TCP Header: Ports

- Identify local end-points
- Beyond 1024, each host decides how to allocate



T-557, PD-376 to 377 27

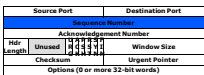
TCP Connections

- Identified by
 - Source Port
 - Source IP address
 - Destination Port
 - Destination IP address
- This is the demux key
- Figure to be drawn in class

T-560 28

TCP Header: Sequence Number

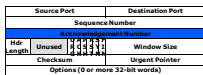
- 32 bits long
- Also known as SSN
 - Send-sequence num.
- Initial value negotiated upon connection
- Future values – byte offset in stream (minus initial value)
- Example – first segment, SSN = n
Number of bytes in first segment = m
Second segment SSN = n+m
- Every byte is numbered



T-558, PD-377 29

TCP Header: Acknowledgement Number

- 32 bits long
- Also known as RSN
 - Receive-sequence number
- Acknowledges receipt of prior bytes from peer entity
- Specifies number of next byte expected (i.e. the next SSN)



T-558, PD-377 30

TCP Header: Header Length

- 4 bits long
- Specifies number of 32 bit words in header
- Usually 5 (no Options fields)
- Next: 6 bit Unused field

Source Port	Destination Port
Sequence Number	Acknowledgement Number
Header Len	Window Size
Checksum	Urgent Pointer
Options (0 or more 32-bit words)	

T-558 31

TCP Header: Flags

- Six one-bit flags
- URG = urgent pointer field valid
 - Urgent Pointer is byte offset to where urgent data can be found
- ACK = acknowledgement num field valid
 - i.e. this segment contains an acknowledgement
- PSH = this is pushed data
 - "Please don't buffer it on receipt" – deliver to application ASAP

Source Port	Destination Port
Sequence Number	Acknowledgement Number
Header Len	Window Size
Checksum	Urgent Pointer
Options (0 or more 32-bit words)	

T-558, PD-377 to 378 32

TCP Header: Flags (cont.)

- RST = reset connection
 - Connection is aborted
 - Also used to refuse connection
- SYN = synchronize sequence numbers
 - Used during connection
 - SYN=1, ACK=0 is REQUEST_CONNECTION
 - SYN=1, ACK=1 is ACCEPT_CONNECTION
- FIN = finish
 - Sender has no more data to transmit
 - Release connection

T-558 to 559, PD-378 33

TCP Header: Advertised Window

- Available buffer size for receiving data
- Sent with acknowledgement
- Associated with Sliding-Window Protocol (covered later)

Source Port	Destination Port
Sequence Number	Acknowledgement Number
Header Len	Window Size
Checksum	Urgent Pointer
Options (0 or more 32-bit words)	

T-559 34

TCP Header: Checksum

- As for UDP checksum
- Includes
 - Header
 - Data
 - Pseudo header (Src + Dst IP addresses + length)
- Compulsory (Optional for UDP)

Source Port	Destination Port
Sequence Number	Acknowledgement Number
Header Len	Window Size
Checksum	Urgent Pointer
Options (0 or more 32-bit words)	

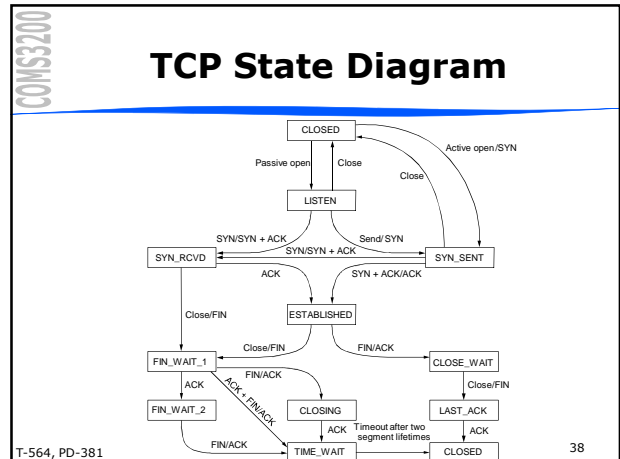
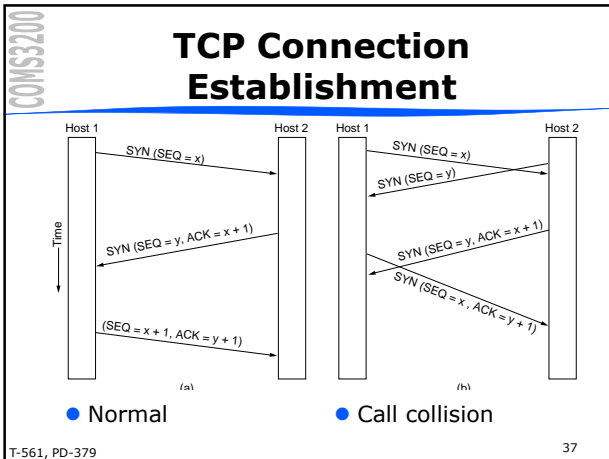
T-559 35

TCP Header: Options

- Optional
 - Padded out to 32 bit boundary
- Examples:
 - Negotiate max segment size during connection
 - Scaled window
 - Shift window size field value 14 bits left
 - Selective repeat
 - Negative acknowledgement (NAK)

Source Port	Destination Port
Sequence Number	Acknowledgement Number
Header Len	Window Size
Checksum	Urgent Pointer
Options (0 or more 32-bit words)	

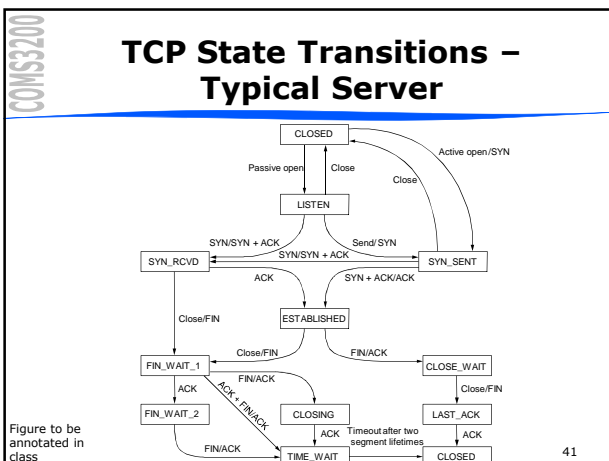
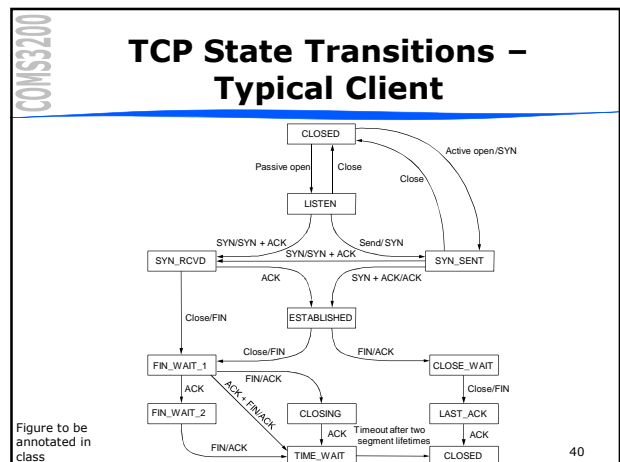
T-559 to 560 36



Connection release

- What communication takes place to close a TCP connection?
- As TCP State Diagram shows - connection is gracefully released without data loss
 - Connection is full duplex - each side has to send FIN which needs to be ACK-ed

39



Sliding Window Protocol

- Sliding window protocols
 - Used for flow control
 - Mostly used at transport and data link layers
 - Transport layer – dynamic window
 - Data Link layer – static window
- Issue will be revisited in lecture 5

42

Sliding Window

- Allow multiple outstanding (un-ACKed) messages
- Upper bound on un-ACKed messages, called *window*

43

Sliding Window

Sending application

Receiving application

- Sending side
 - $LastByteAacked \leq LastByteSent$
 - $LastByteSent \leq LastByteWritten$
 - buffer bytes between $LastByteAacked$ and $LastByteWritten$
- Receiving side
 - $LastByteRead < NextByteExpected$
 - $NextByteExpected < = LastByteRcvd + 1$
 - buffer bytes between $LastByteRead$ and $LastByteRcvd$

44

Flow Control

- Receiving side – buffer: $MaxRcvBuffer$
 - $LastByteRcvd - LastByteRead \leq MaxRcvBuffer$
 - $AdvertisedWindow = MaxRcvBuffer - (LastByteRcvd - LastByteRead)$
- Sending side – buffer: $MaxSendBuffer$
 - $LastByteSent - LastByteAacked \leq AdvertisedWindow$
 - $EffectiveWindow = AdvertisedWindow - (LastByteSent - LastByteAacked)$
 - $LastByteWritten - LastByteAacked \leq MaxSendBuffer$
 - block sender if try to transmit y bytes and $(LastByteWritten - LastByteAacked) + y > MaxSendBuffer$
- Always send ACK in response to arriving data segment
- Persist with sending 1 byte of data even when $AdvertisedWindow = 0$

45

TCP flow control example

46

Delayed Acknowledgements

- TCP has 500ms delay on acknowledgements and window updates
- Reduces bandwidth waste for applications which send very small segments
- Example:
 - Telnet session

47

Flow Control vs Congestion Control

- What is the difference?
 - Flow control regulates flow of messages between source and destination (related to receiver capacity)
 - Flow control window - window granted by receiver
 - Network is congested when it is overloaded (related to network capacity)
 - Lost packets assumed to mean congestion
 - TCP congestion prevention – slow start, adaptive window and timeout

48

Congestion Control (1)

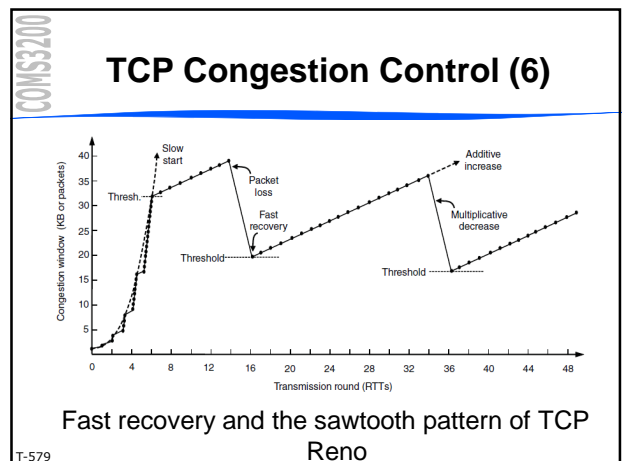
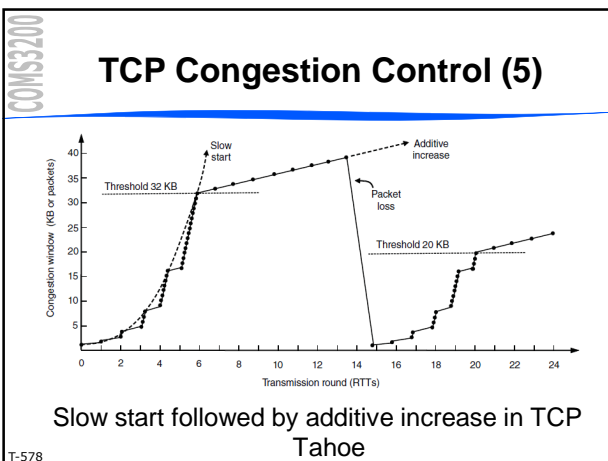
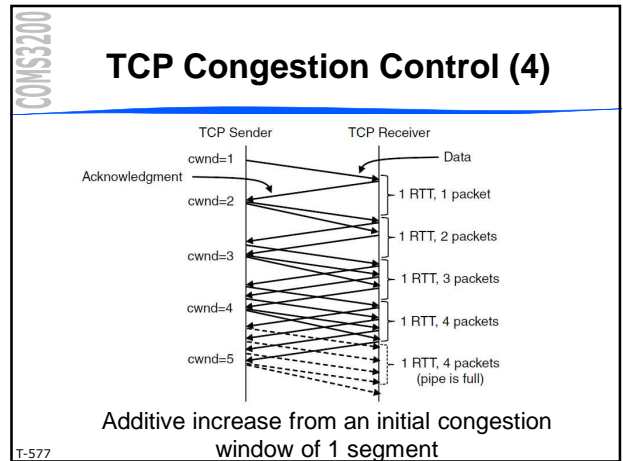
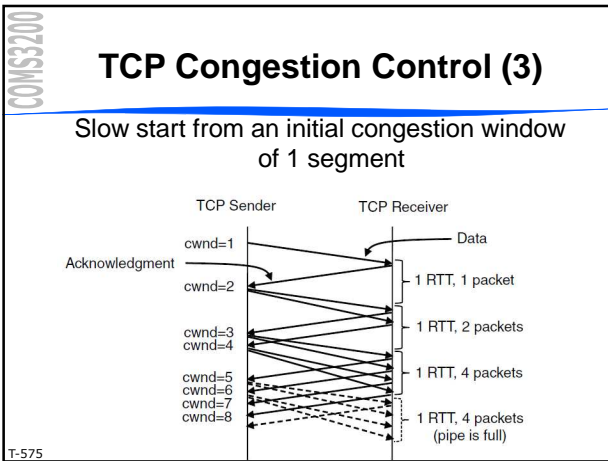
- Slow start
 - TCP sender starts with **one** segment window (effective window)
 - Effective window grows exponentially with each successfully received ACK until it reaches **threshold**
 - After reaching **threshold** effective window grows by one (additive increase)
 - Effective window is limited by flow control window

49

Congestion Control (2)

- If timeout happens
 - Threshold is halved
 - Timeout is adjusted (doubled)
 - Sender returns to slow start

50



Difficulty in setting TCP timer

(a) Probability density of ACK arrival times in the data link layer.
 (b) Probability density of ACK arrival times for TCP.

T-569 55

TCP vs new network technologies

- TCP does not perform well for new technologies
 - High speed long distance network (high bandwidth delay product networks)
 - Range of sequence numbers too small
 - Flow control window too small
 - Slow start takes long time to reach flow control window size
 - Wireless networks attached to the Internet
 - Packet lost due to transmission errors treated as congestion indication
- There are many enhanced versions of TCP

COMS3200 56

Sequence Number Wrap Around

- 32-bit SequenceNum

Bandwidth	Time Until Wrap Around
T1 (1.5 Mbps)	6.4 hours
Ethernet (10 Mbps)	57 minutes
T3 (45 Mbps)	13 minutes
FDDI (100 Mbps)	6 minutes
STS-3 (155 Mbps)	4 minutes
STS-12 (622 Mbps)	55 seconds
STS-24 (1.2 Gbps)	28 seconds

- Maximum Segment Lifetime (MSL) assumed to be 120 seconds
- Enhancement proposal: add timestamp to sequence number

COMS3200 57

Keeping the Pipe Full

- 16-bit AdvertisedWindow
- Assuming 100ms Round Trip Time:

Bandwidth	Delay x Bandwidth Product
T1 (1.5 Mbps)	18KB
Ethernet (10 Mbps)	122KB
T3 (45 Mbps)	549KB
FDDI (100 Mbps)	1.2MB
STS-3 (155 Mbps)	1.8MB
STS-12 (622 Mbps)	7.4MB
STS-24 (1.2 Gbps)	14.8MB

- 16 bits only allows advertised window of 64kB
- TCP extension for larger windows

COMS3200 T-559, PD-389 58

TCP for wireless connections

- Several TCP enhancements proposed
 - e.g. Splitting a TCP connection into two connections (violates end-to-end semantics)

COMS3200 59

Readings

- For Week 3:
 - (Tanenbaum 5th ed.): pg 541-543 (UDP); pg 552-581 (TCP)
 - (Tanenbaum 4th ed.): pg 524-526 (UDP); pg 532-553 (TCP)
- Readings for week 4
 - Physical layer – Chapter 2, especially 2.2 “Guided Transmission Media”, 2.3 “Wireless Transmission”

COMS3200 60