

COMS3200

COMS3200 – Week 5 Data Link Layer

School of Information Technology and Electrical Engineering
The University of Queensland

COMS3200

Data Link Layer

Application Layer (1L)
Transport Layer (1L)
Network Layer (2L)
Link Layer (2L)
Physical Layer (1L)

2

COMS3200

Outline

- Link Layer functionality
 - Framing
 - Error detection
 - Reliable delivery
 - Flow Control

3

COMS3200

Learning objectives

After this week you should understand the role of protocols at the DL layer

- framing
- error detection
 - Parity bits
 - CRC (Cyclic Redundancy Check)
- reliable delivery
 - Automatic Repeat Request (ARQ)
 - stop-and-wait
 - Go-back-N
 - Selective-repeat
- flow control
 - Prevent slow receiver being swamped by fast sender

4

COMS3200

Terminology

- In TCP/IP model, this is called Link Layer.
- In OSI Model, this is called Data Link Layer.
- We (& Tanenbaum) use Link Layer, Data Link Layer, DL Layer interchangeably.
- Others use Network Interface Layer.
- Often Link Layer is divided into Medium Access Control (MAC) sub-layer (bottom) and Logical Link Control (LLC) sublayer (top).
- We will do MAC sub-layer next week.

5

COMS3200

Link Layer Functionality

- Link Layer accesses services of Physical Layer
 - Transmission of raw bits over wire
- Link Layer provides services to Network Layer
 - Delivery of frames between adjacent nodes
 - (i.e. there is a direct physical link between the nodes)

6

Link Layer Functionality

- Link Layer has to overcome deficiencies of Physical Layer
 - Transmission Errors
 - Physical channel can invert bits or garble frames
- Link Layer provides
 - Framing
 - Break up bit stream into discrete frames
 - Error detection
 - Reliable delivery (optional)
 - Flow Control (optional)

Link layer protocol

PDU - Protocol Data Unit

Packets and Frames

Relationship between packets and frames.

Link Layer Protocol

Framing

- Break sequence of bits into frames
 - clearly marks beginning and end of frame
- Typically implemented by network adaptor (network card)
- Problem:
 - How can we detect beginning and end of a frame?

Framing Approaches

- Sentinel-based
 - Character-based (byte stuffing)
 - Binary (bit stuffing)
- Counter based
- Clock-based
- Coding violation

Sentinel-based Framing

- Character based protocols
 - Use sequence of ASCII characters to mark beginning and end of frames
 - OK for text-based payload
 - Problem if payload is binary data
 - DLE STX and DLE ETX can occur 'accidentally' in payload
 - Incorrect detection of frame start/end

13

Character Stuffing

- For each 'accidental' DLE in payload, the sender's data link layer inserts another DLE in front
- Framing DLE STX or DLE ETX can be distinguished from accidental DLE STX or DLE ETX through absence or presence of double DLE
 - DLE in payload is always doubled
- The receiver's link layer removes additional DLE before passing data to the network layer

14

Bit Stuffing

- Sentinel-based framing for binary protocols
 - Use special bit pattern (flag byte) to mark start and end of frame
 - 01111110 (e.g., HDLC, SDLC, PPP)
 - Problem: What if flag byte occurs in payload?
 - solution: *bit stuffing*
 - sender: insert 0 after five consecutive 1s for payload data
 - receiver: delete 0 that follows five consecutive 1s before passing data to network layer

15

Bit Stuffing

(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

(b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0

↑
Stuffed bits

(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Bit stuffing
 (a) The original data.
 (b) The data as they appear on the line.
 (c) The data as they are stored in receiver's memory after destuffing.

If 01111101 appears in payload → 011111001
 Whenever receiver sees 5 1s followed by a 0, it automatically destuffs it.

16

Counter-based Framing

- include payload length in header
 - e.g., DDCMP (DECnet)
- problem: count field can get corrupted
- Only partial solutions to problem → counter-based framing is not used in newer protocols

17

Framing approaches (cont)

- Clock-based
 - each frame is 125µs long
 - e.g., SONET: Synchronous Optical Network
 - Requires exact synchronisation
- Coding violation (used in Ethernet in conjunction with flag byte)
 - Different signal than used for "1" or "0"
 - Every bit is encoded as a pair of "physical bits" (Manchester encoding)
 - high - low = 1
 - low - high = 0
 - E.g. High - high and low- low are not used for data and can be used to mark beginning of frame

18

Transmission Error Detection

EDC= Error Detection and Correction bits (redundancy, checksum)
 D = Data protected by error checking, may include some header fields

← d data bits →
 D | EDC
 (bit-error prone link)
 datagram
 datagram
 all bits in D' OK ?
 N detected error

K-5.4 19

Parity Checking

- Even parity: Set parity bit so that total number of 1s is even
- Odd parity: Set parity bit so that total number of 1s is odd
- Can detect single bit errors and any odd number of bit errors

d data bits Parity bit (even) Transmission error
 01001100 1 → 01011100 1
 Receiver counts 1s → error detected!

K-5.5 20

Parity checking

Two dimensional parity

	$d_{1,1}$...	$d_{1,j}$	$d_{1,j+1}$	row parity
	$d_{2,1}$...	$d_{2,j}$	$d_{2,j+1}$	
	
	$d_{i,1}$...	$d_{i,j}$	$d_{i,j+1}$	
column parity	$d_{i+1,1}$...	$d_{i+1,j}$	$d_{i+1,j+1}$	

K-5.6 21

Parity checking

Two dimensional parity - example

101011	101011	
111100	101100	parity error
011101	011101	
01010	01010	

no errors
 parity error
 correctable single bit error

How many bit errors can be detected?

K-5.6 22

Error Detection Codes

- Add r bits of redundant (checksum) data to an m -bit message
 - Total message is of length $n = m + r$
 - The n bits are called a *code word*
 - m/n is called the *code rate*
 - The lower m/n , the higher the overhead of the code
 - want $r \ll m$
 - e.g., $r = 32$ and $m = 12,000$ (1500 bytes)

$$n = m + r \text{ (bits)}$$

m data bits | r checkbits

K-5.6 23

Error Detection Codes

- $n = m + r$
- Number of valid code words: 2^m (r bits are redundant, i.e. determined by m bits of data)
- Number of n -bit words: 2^n
- Idea of error detection: Transmission error will result in an invalid code word, i.e. with wrong checksum

Code word space:

n-bit words | valid code words

K-5.6 24

COMS3200

Hamming Distance

- Hamming distance: The number of bit positions in which two words differ
 - $d(00101, 00010) = 3$
 - $d(00000, 11111) = 5$
- If two code words have a Hamming distance d , it takes d single-bit errors to convert one into the other
- Hamming Distance of a Code is the minimum Hamming distance between any two code words
 - A Code with Hamming Distance d can detect up to $d-1$ single-bit errors

25

COMS3200

Example

- $m=2, r=2$
- The r checkbits are simply a copy of the m data bits
- Data words: 00, 01, 10, 11
- Codewords: 0000, 0101, 1010, 1111
- n -bit words: 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111
- What's the rate of the code?
- What's the Hamming distance of the Code?
- How many single-bit errors can it detect?
- Is this code efficient? (compare overhead with robustness to error)

26

COMS3200

Cyclic Redundancy Check

- Most commonly used error detection code, e.g. Ethernet
 - "Cyclic Redundancy Code" (CRC)
 - "Polynomial Code"
- Represent m -bit message M as $m-1$ degree polynomial $M(x)$
 - $M = 1001 \rightarrow M(x) = 1*x^3 + 0*x^2 + 0*x^1 + 1*x^0 = x^3 + 1$
 - $M = 10011010 \rightarrow M(x) = x^7 + x^6 + x^4 + x^3 + x^1$

27

COMS3200

CRC

- Sender and Receiver must first agree on a generator polynomial $G(x)$ of degree r .
- m -bit data correspond to polynomial $M(x)$ of maximum degree $m-1$
- Total codeword (data + CRC bits) corresponds to polynomial $T(x)$ of max degree $m+r-1$

28

COMS3200

CRC Idea

- Idea is to append a r -bit checksum to the end of the m -bit frame so that the polynomial $T(x)$ represented by the checksummed frame is divisible by the generator $G(x)$.
- Receiver checks if $T(x)/G(x) = 0$
- Compute checksummed frame $T(x)$:
 - $G(x)$ is of degree r . Append r zero bits to the low-order end of the frame. Frame now contains $m+r$ bits and corresponds to polynomial $x^r M(x)$
 - Compute the remainder of $x^r M(x) / G(x)$
 - Subtract the remainder from $x^r M(x) \rightarrow T(x)$
 - $T(x)$ will be evenly divisible by $G(x)$
 - In general, if we subtract remainder from dividend, the result will always be evenly divisible by divisor!
 - $T(x)/G(x)=0$: no error
 - $T(x)/G(x) \neq 0$: transmission error

29

COMS3200

Polynomial Arithmetic modulo 2

- Defined by branch of mathematics called Field Theory
- Addition and subtractions are both done via XOR
- Use normal long division (in binary) to divide by $G(x)$
- Use XOR for addition **and** subtraction (no carries and borrows)
- Can be implemented efficiently in Hardware

Addition, Subtraction:

$$\begin{array}{r} x^3 + x + 1 \rightarrow 1011 \\ + - x^2 + 1 \rightarrow 0101 \\ \hline x^3 + x^2 + x \leftarrow 1110 \end{array}$$

Division by $G(x)$:

$$x^7 + x^4 + x^3 + x + 1 : x^4 + 1 \rightarrow$$

$$10011011 : 10001 = 1001$$

A divisor is said "to go into" a dividend if the dividend has at least as many bits as the divisor:
Example:
 $1000 : 1111 = 1$

$$\begin{array}{r} 10011011 \\ \underline{10001} \\ 00000 \\ \underline{0001} \\ 00000 \\ \underline{0001} \\ 0111 \\ \underline{0111} \\ 0000 \end{array}$$

10 (Remainder)

30

CRC Example

- $G(x) = x^3 + 1 \rightarrow 1001$ ($r=3$)
- Data $M(x) = x^4 + x^2 + 1 \rightarrow 10101$
- $M(x)x^3 = x^7 + x^5 + x^3 \rightarrow 10101000$
- Calculate Remainder of $M(x)x^3/G(x)$:

$$\begin{array}{r} 10101000 : 1001 = 10111 \\ \underline{1001} \\ 111 \\ \underline{0000} \\ 1110 \\ \underline{1001} \\ 1110 \\ \underline{1001} \\ 1110 \\ \underline{1001} \\ 111 \\ \end{array}$$

111 (Remainder)

- Checksummed Frame:
 - Subtract Remainder from dividend
 - $10101000 - 111 = 10101111$
 - (remember: adding and subtracting is the same)
- Check if $10101111 : 1001 = 0$?

31

CRC Error Detection Capabilities

- Transmission Error: $T(x) + E(x)$
 - Bit errors can be considered as addition of error polynomial $E(x)$
- At receiver:
 - $[T(x) + E(x)]/G(x) = T(x)/G(x) + E(x)/G(x) = E(x)/G(x)$
- Error goes undetected only if error polynomial $E(x)$ happens to have $G(x)$ as a factor
- Choice of $G(x)$ is crucial (Tanenbaum 3.2.2)
- Good generator polynomials of various degrees have been standardised:
 - CRC-16
 - CRC-32 (used in Ethernet)

32

Internet Checksum Algorithm

- Used for IP (header), TCP (data + header), UDP (data + header, optional)
- For IP, checksum needs to be re-computed at each hop \rightarrow checksum algorithm needs to be efficient (implementation in software)
- View message as a sequence of 16-bit integers; sum using 16-bit ones-complement arithmetic (bits inverted); take ones-complement of the result.
- Simple but not as powerful as CRC. E.g. cannot detect
 - Reordering of message blocks
 - Inserting or deleting zero-valued bytes
 - Multiple errors that cancel each other out

33

Internet Checksum Algorithm

```

u_short
cksum(u_short *buf, int count)
{
    register u_long sum = 0;
    while (count--)
    {
        sum += *buf++;
        if (sum & 0xFFFF0000)
        {
            /* carry occurred, so wrap around */
            /* Trick to compute 1's complement sum
             on two's complement architecture*/
            sum &= 0xFFFF;
            sum++;
        }
    }
    return ~(sum & 0xFFFF);
}

```

34

Error Correction

- Much more complex
- FEC (Forward Error Correction)
- Used when there is no time for retransmission (e.g. spaceships) and for WiMax
- Need to add more redundancy \rightarrow bigger overhead
- E.g. Used on CDs (Reed-Solomon Code)
- Error Correction not covered in COMS3200

35

Link Layer Functionality

- Link Layer functionality
 - Framing
 - Error detection
 - **Reliable delivery**
 - (partly covered in Lecture on TCP)
 - Flow Control

36

COMS3200

Reliable delivery

- Reliability is typically discussed at the Link Layer
- Is also implemented at higher layers (TCP)
- Most Link layer protocols are unreliable (e.g. Ethernet)
 - Reliable delivery for Link layers makes sense for highly unreliable transmission media e.g. wireless links (802.11)

37

COMS3200

Reliable delivery

- Physical layer can invert bits or garble frames
- Reliable delivery protocols use
 - acknowledgements
 - timeouts

38

COMS3200

Reliable delivery

- What happens if frames or ACKs are lost?
- Solution:
 - Setup **timer**
 - When timer is off, **retransmit** the packet
- Often called ARQ (*automatic repeat request*)

39

COMS3200

Acknowledgements & Timeouts

40

COMS3200

Acknowledgements

- Acknowledgements can be positive (ACK) or negative (NAK)
- ACK means that a frame has been received
- NAK means that a frame has not been received correctly (error detection)

41

COMS3200

Stop-and-Wait Protocol

- Sender sends one packet/frame, then waits for receiver's response

42

Stop-and-Wait

- What could have happened if sender does not receive an ACK before timeout?
 - Frame could have been lost
 - ACK could have been lost
- Sender cannot tell what happened
 - resends frame
 - In case 2, this results in a duplicate frame
 - Problem if passed up to the application (e.g. file transfer)
 - Solutions?

The diagram shows two scenarios. In the first, a frame is sent from the sender to the receiver, but it is lost (indicated by a red 'X'). A vertical double-headed arrow labeled 'Time out' is shown on the sender side. In the second scenario, a frame is sent and received, but the ACK is lost. The sender's 'Time out' period expires, and it resends the frame. The receiver then receives a 'Duplicate Frame!'.

Stop and Wait

Handling duplicates:

- sender adds *sequence number* to each frame
- Each ACK contains sequence number of frame it acknowledges
- Receiver can detect duplicate frames and discard them

Stop and Wait example

The diagram shows a successful transaction. The sender sends 'pkt 0', which is received by the receiver. The receiver sends 'ack 0', which is received by the sender. The sender then sends 'pkt 1', which is received by the receiver. The receiver sends 'ack 1', which is received by the sender. Finally, the sender sends 'pkt 0' and the receiver sends 'ack 0'.

Stop and Wait discussion

Sender:

- seq # added to frame
- two seq. #'s (0,1) will suffice
 - Only ambiguity is between a frame and its immediate successor/predecessor
- must check if received ACK/NAK corrupted

Stop and Wait discussion

Receiver:

- must check if received frame is duplicate
 - Must remember which sequence # to expect next (state)
- note: receiver *cannot* know if its last ACK/NAK received by sender or not

Stop and Wait example

(a) operation with no loss: The sender sends 'pkt 0', the receiver receives it and sends 'ACK 0'. The sender receives 'ACK 0' and sends 'pkt 1'. The receiver receives 'pkt 1' and sends 'ACK 1'. The sender receives 'ACK 1' and sends 'pkt 0'. The receiver receives 'pkt 0' and sends 'ACK 0'.

(b) lost packet: The sender sends 'pkt 0', the receiver receives it and sends 'ACK 0'. The sender receives 'ACK 0' and sends 'pkt 1'. The packet is lost (indicated by a red 'X' and 'loss'). A 'timeout' occurs on the sender side, and it resends 'pkt 1'. The receiver receives 'pkt 1' and sends 'ACK 1'. The sender receives 'ACK 1' and sends 'pkt 0'. The receiver receives 'pkt 0' and sends 'ACK 0'.

Stop and Wait example

(c) lost ACK

(d) premature timeout

K-3.16 49

Stop-and-Wait performance

- Problem: keeping the channel full
 - Send a small packet
 - Wait for ACK
 - Send next packet ...
- Very inefficient for some channels
 - A lot of waiting involved

COMS3200 50

Stop-and-Wait performance

f: frame size in bits
 b: data rate of channel [bps]
 d: propagation delay [s] ($d \approx 5\text{ms per } 1000\text{km}$)
 u: line utilisation

- Total time to send a frame (Assume that ACK frame is very small):
 - $d + f/b + d$
 - Transmission time: f/b
- Line utilisation:
 - out of total time required to send a frame, what's the percentage that is actually used to transmit data? (How full is the channel?)
 - $u = f/b / (2d + f/b) = f / (2d \cdot b + f)$
 - $\rightarrow u$ is small for links with high bandwidth delay product

Example:
 • $f = 1000$ bits
 • $b = 1\text{Mbps}$
 • $d = 20$ ms
 $u = 1\text{ms}/41\text{ms} = 2.4\%$

Solution?

51

Transmission and Propagation Delay Applet

- http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/transmission/delay.html

COMS3200 52

Sliding Window

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

53

Sliding Window: sender

- Assign sequence number to each frame (seqNum)
- Maintain three state variables:
 - send window size (sWS)
 - last acknowledgment received (LAR)
 - last frame sent (LFS)
- Maintain invariant: $LFS - LAR \leq sWS$

- Slide the Window (Ring buffer)
 - Advance LAR when ACK arrives
 - Advance LFS when new frame is sent
- Buffer up to sWS frames (for potential retransmission)

54

Sliding Window: receiver

- Maintain three state variables
 - receive window size (*RWS*)
 - largest frame acceptable (*LFA*)
 - last frame received (*LFR*)
- Maintain invariant: $LFA - LFR \leq RWS$

- Frame *seqNum* arrives:
 - Accept if within window
- Slide the Window
 - Advance **LFR and LFA** when frame LFR+1 (at lower edge of window) is received

55

Sliding Window Protocols

- Go-Back-N
 - Receiver Window Size = 1
 - Receiver only accepts frames in order
 - If frame is lost, retransmit that frame and all other frames sent thereafter
- Selective Repeat
 - Receiver Window Size > 1
 - If frame is lost, only retransmit missing frame

56

Go Back N

- If error occurs, go back to oldest frame that was sent and is not Ack'ed, and start retransmitting from there

T-234 57

Go Back N

- Receiver: out-of-order packets are discarded
- ACK(*n*): acknowledges all frames up to, including seq # *n* ("cumulative ACK")
 - E.g receiver sends ACK1, ACK2 and ACK3
 - It does not matter if ACK1 or 2 is lost, since ACK3 implies ACK1 and ACK2
- timer for each sent frame
- timeout(n)*: retransmit frame *n* and all higher seq # frames in window

58

Go Back N example

K-3.21 59

Sequence Number Space

- SeqNum** field is finite; sequence numbers wrap around
- Example:
 - 3-bit field sequence numbers will range 0,...,7 (**MaxSeqNum=7**)
 - Assume SWS = nbr of distinct sequence numbers = $\text{MaxSeqNum} + 1 = 8$
 - Scenario 1:
 - sender transmits, but all ACKs are lost
 - sender retransmits 0..7
 - receiver expecting new 0..7, but receives second incarnation of 0..7 and treats frames as new ones
- $\text{SWS} < \# \text{ distinct sequence numbers}$
- $\text{SWS} \leq \text{MaxSeqNum}$

60

Last lecture...

- Recap
 - DLL functionality
 - Framing
 - Bit stuffing example
 - Error detection
 - Reliability
 - ARQ
 - » Stop-and wait → channel utilisation → Applet
 - » Go-back-N

61

Go Back N Discussion

- Small receiver window, only small buffer required at receiver
- Wastes bandwidth by retransmitting already successfully transmitted frames
- Applet: http://media.pearsoncmg.com/aw/aw_kuruse_network_2/applets/go-back-n/go-back-n.html

62

Selective Repeat

- Receiver Window > 1
- receiver *individually* acknowledges all correctly received frames
 - buffers correctly received out-of-order frames, for eventual in-order delivery to upper layer
- Sender only resends frames which were not Acknowledged
 - sender timer for each unACKed frame

63

Selective Repeat

- Retransmission can be triggered by
 - Time out
 - NAK
 - Duplicate ACK

(b)

64

Selective Repeat: Sender and Receiver Window

(a) sender view of sequence numbers

(b) receiver view of sequence numbers

K-3.22

65

Selective Repeat Example

(b)

K-3.25

66

Window size versus seq. number

- Example:
 - seq #'s: 0, 1, 2, 3
 - window size=3

K-3.26 (a) 7

Window size versus seq. number

- Rule: There should be no overlap in sequence numbers between before and after window is advanced (in case of loss of ACKS)
- $SWS \leq (MaxSeqNum + 1) / 2$

Sender	Receiver
12345678	12345678
12345678	12345678

- Receiver accepts frame 5-8. If all acks get lost and sender resends frames 1-4, the receiver won't accept any of those duplicate frames.
- Intuitively, SeqNum "slides" between two halves of sequence number space
- (For Go Back N max $SWS = MaxSeqNum$)

COMS3200 68

Selective repeat demo

- http://media.pearsoncmg.com/aw/aw_kurose_network_4/applets/SR/index.html

COMS3200 69

Sliding Window Protocol Summary (1)

- Sliding window protocols are used by
 - Reliable data link layer protocols
 - e.g. in wireless networks
 - Reliable transport layer protocols
 - e.g. TCP

COMS3200 70

Sliding Window Protocol Summary (2)

- Go-back-N
 - Used in "standard" TCP
 - Requires only small buffer at receiver
 - Retransmits successfully transmitted frames - > waste of bandwidth
- Selective repeat
 - Newer versions of TCP support optional SACK (selective ACK)
 - More effective use of bandwidth
 - Sender only retransmits missing frames
 - Requires larger buffer at receiver, Receiver Window > 1

COMS3200 71

Link Layer Functionality

- Link Layer functionality
 - Framing
 - Error detection
 - Reliable delivery
 - Flow Control**

COMS3200 72

COMS3200

Flow Control

- A receiver has limited resources in terms of buffers, and
- Sender may be faster than receiver
- Sender should not flood receiver

Solution?

73

COMS3200

Flow Control

- Solution:
 - Sliding Window Protocols
 - Sender can send a certain amount of data (window size) before it has to wait for the receiver to catch up
 - See Lecture on TCP

74

COMS3200

Data delivery

- Three possible services provided by Link Layer to Network Layer
 - Unacknowledged connectionless service
 - no error recovery, suitable for low error rate channels
 - Ethernet

75

COMS3200

Data delivery

- Acknowledged connectionless service
 - suitable for unreliable channels
 - 802.11
- Acknowledged connection-oriented service
 - suitable for WAN subnets connected by point-to-point leased lines or ATM network

76

COMS3200

Existing Data Link Layer protocols (1)

- Most of DLL protocols are bit-oriented and use bit stuffing
 - exceptions: old protocols like IBM BSC or SLIP or PPP on analog lines (use character stuffing)
- Best known DLL protocol is HDLC (High-level Data Link Control) - used in X.25 networks
 - it can work as Go Back N or Selective repeat
 - it can use 3 or 7 bits sequence number

77

COMS3200

Existing Data Link layer protocols (2)


- LAN DLL protocols are HDLC-like (use Go Back N for error and flow control)
- New very reliable transmission media (e.g. fiber optics) brought a change to DLL protocols
 - due to low error rate, recovery from errors (lost frames) can be left to the transport layer (e.g. approach used in Frame Relay)
- There are still unreliable media which require reliable protocols (e.g. wireless networks, 802.11)

78

COMS3200

Reliable Link Layer

- What's the point of implementing reliability at the link layer if transport layer implements it anyway?
- Example: Wireless Network, 802.11
 - Local error control prevents the retransmission of many packets end-to-end



79

COMS3200

Summary

- DLL is responsible for delivery of packets to adjacent nodes
 - DLL `frames' packets
 - DLL detects transmission errors
 - DLL may provide reliability of transmission and flow control
- Reliability (ack, timeout, window) and flow control (window) techniques:
 - stop&wait, Go-Back-N, Selective Repeat

80

COMS3200

Readings

- Tanenbaum: 3.1-3.6
- Next week: Medium Access Sublayer
 - Tanenbaum: 4.1, 4.2 (without 4.2.4), 4.3.1, 4.3.4
 - (Same chapters in 4th edition)

81