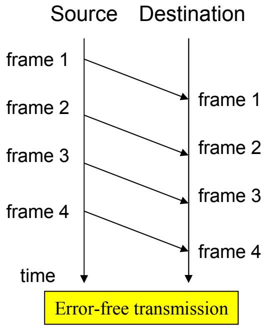
King Fahd University of Petroleum & Minerals
Computer Engineering Dept

COE 540 –Computer Networks
Term 072
Dr. Ashraf S. Hasan Mahmoud
Rm 22-148-3
Ext. 1724
Email: ashraf@kfupm.edu.sa

Lecture Contents

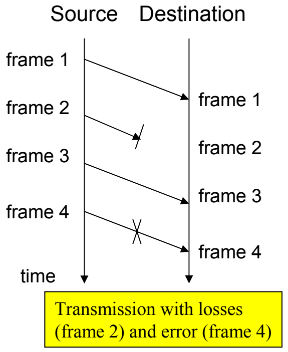
1. ARQ: Retransmission Strategies
    a. Stop-and-Wait
    b. Go Back n ARQ
    c. Selective Repeat ARQ
2. Examples: ARPANET ARQ
3. Framing
    a. Character-Based Framing
    b. Bit Oriented Framing
4. Standard DLCs

Issues Frame Transmission



Error-free transmission

Models of Frame Transmission



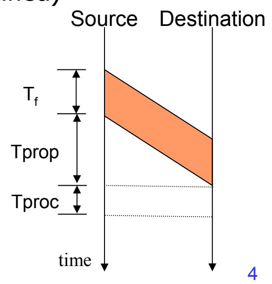
Transmission with losses (frame 2) and error (frame 4)

- The destination has a limited buffer space. How will the source know that destination is ready to receive the next frame? Need for flow control
- Two types of damaged frames: erroneous frame or frame lost!
- In case of errors or lost frame, the source need to retransmit frames – i.e. a copy of transmitted frames must be kept. How will the source know when to discard copies of old frames?
- Etc.

Issues Frame Transmission

- A scheme to ensure that transmitter does not overwhelm receiver with data
- Transmission of one frame:
  - T_f: time to transmit frame
  - Tprop: time for signal to propagate
  - Tproc: time for destination to process received frame – small delay (usually ignored if not specified)
- Tproc may be ignored if not specified

Very important representation



Issues Frame Transmission (2)

- Typical frame structure:
  - SN – sequence number for the packet being transmitted
  - RN – sequence number for the NEXT packet in the opposite direction
  - Packet – payload
  - CRC – See previous set of notes
- Piggybacking

	SN	RN		Packet	CRC

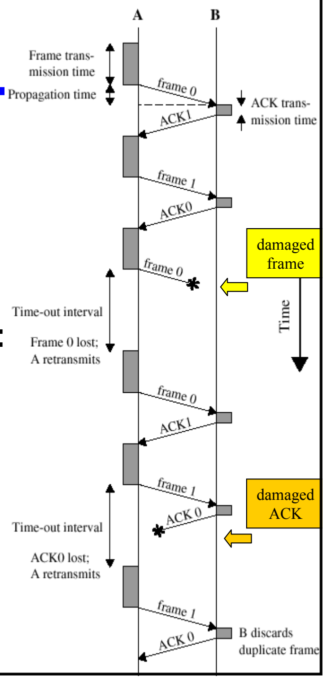
What is ARQ?

- Def: to detect frames in error and then request the transmitter to repeat the erroneous frames
  - Using ARQ, systems can automatically request the retransmission of missing packets or packets with errors.
- Error Control:
  - ARQ
  - Forward Error Correction – Def = ?
- ARQ Algorithms Figures of Merit
  - Correctness (i.e. only one packet released to upper layer)
  - Efficiency (i.e. throughput)
- Three common schemes
  - Stop & Wait
  - Go Back N
  - Selective

Stop-and-Wait Algorithm

- The simplest ARQ algorithm!
- Operation Rules:
  - Algorithm at node A for A-to-B transmission:
    1. Set the integer variable SN to 0
    2. Accept a packet from the next higher later at A; if no packet is available, wait until it is; assign number SN to the new packet
    3. Transmit the SNth packet in a frame containing SN in the sequence number field
    4. If an error-free frame is received from B containing a request number RN greater than SN, increase SN to RN and go to step 2, if no such frame is received is received within some finite delay, go to step 3
  - Algorithm at node B for A-to-B transmission
    1. Set the integer variable RN to 0 and then repeat step 2 and 3 forever
    2. Whenever an error-free frame is received from A containing the sequence number SN equal to RN, release the received packet to the higher later and increment RN
    3. At arbitrary times, but within bounded delay after receiving any error free data from A, transmit a frame to A containing RN in the request number field.
- The textbook provides an *informal proof* for the correctness of the above algorithm:
  - Liveness: can continue for ever to accept new packets at A and release them to B
  - Safety: never produces an incorrect result (i.e. never releases a packet out of the correct order to the higher layer)

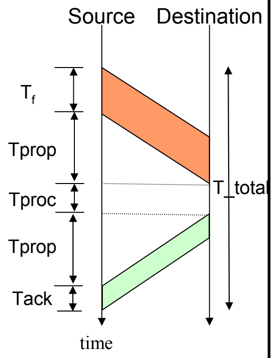
Modulo 2 Stop-and-Wait ARQ



- Uses Modulo 2 sequence numbers (SN and RN)
  - Both frames and ACKs are numbered
- Two types of errors:
  1. Frame lost or damaged – *Solution*: timeout timer
  2. Damaged or lost ACK – The timeout timer solves this problem

Stop-and-Wait Protocol: Efficiency

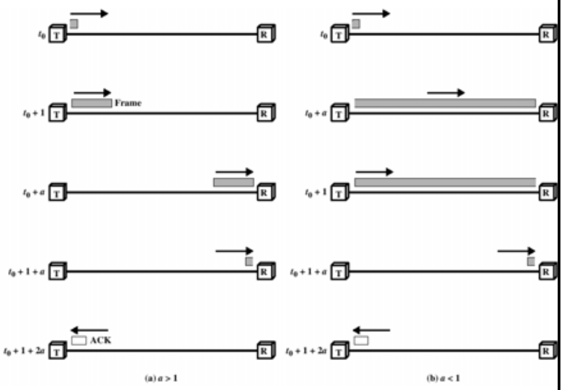
- After every frame, source must wait till acknowledgment → Hence link propagation time is significant
- Total time to for one frame:
  T_total = Tf + 2Tprop + Tproc + Tack
  if we ignore Tproc and Tack (usually very small)
  T_total = Tf + 2Tprop
- Link utilization, U is equal to
  U = Tf/ (T_total), or
  = 1 / (1+2(Tprop/Tf)) = $\boxed{1 / (1 + 2 a)}$
  where a = Tprop/Tf = length of link in bits
- If a < 1 (i.e. Tf > Tprop – when 1st transmitted bit reaches destination, source will still be transmitting → U is close 100%
- If a > 1 (i.e. Tf < Tprop – frame transmission is completed before 1st bit reaches destination →U is low



Stop-and-Wait Protocol: Efficiency (2)

- Remember: a = Tprop/Tf = length of link in bits

- If a < 1 (i.e. Tf > Tprop – when 1st transmitted bit reaches destination, source will still be transmitting → U is close 100%
- If a > 1 (i.e. Tf < Tprop – frame transmission is completed before 1st bit reaches destination →U is low
- Stop-and-Wait is efficient for links where a << 1 (long frames compared to propagation time)



Stop-and-Wait Protocol: Efficiency With Errors (3)

- Assume a frame is in error with probability P
- Therefore, average utilization can be written as

U = Tf / (Nr x T_total)

- Nr is the average number of transmissions of a frame, while T_total is equal to Tf + 2Tprop.
- For stop-and-wait, Nr is given by

Nr = E[no of transmissions] = Σ i x Prob[i transmissions]
= Σ i x P^{i-1}(1-P)
= 1/(1-P)

- Therefore, utilization is given by

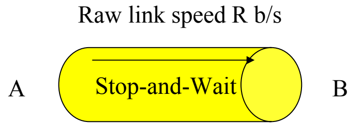
$\boxed{U = (1-P)/(1+2a)}$

Identities:
Σ (X^{i-1},i=1,∞) =1/(1-X) for -1<X<1
Σ (iX^{i-1},i=1,∞) =1/(1-X)^2 for -1<X<1

- Note that for P = 0 (i.e. error free), the expression reduced to the previous result!

Important Performance Figures

- Utilization (U) – fraction of time the link is used for transmitting data
- Throughput (b/s) – effective b/s as experienced by user data
  - Throughput = R * U (b/s)
- Throughput (frame/s) – average data frames per second the link is supporting
  - Throughput = 1/T_total (frame/sec)
  - = R*U/data_frame_size (frame/sec)



Utilization = ?
Throughput = ?

Sliding Window Protocol

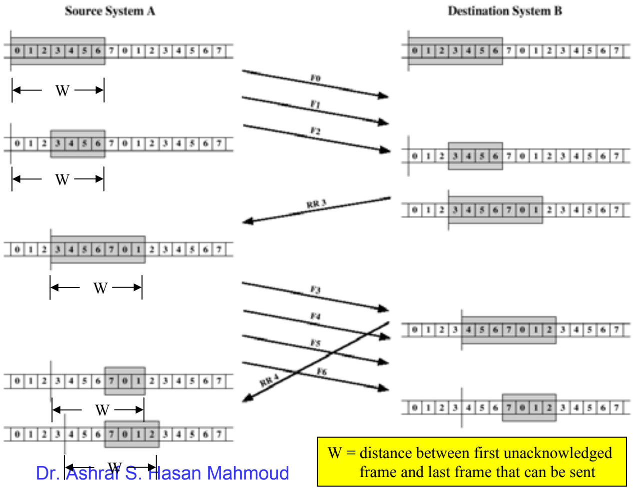
- Stop-and-Wait can be very inefficient when a > 1
- Protocol:
  - Assumes full duplex line
  - Source A and Destination B have buffers each of size W frames
  - For k-bit sequence numbers:
    - Frames are numbered: 0, 1, 2, …, 2^k-1, 0, 1, … (modulo 2^k)
    - ACKs (RRs) are numbered: 0, 1, 2, …, 2^k-1, 0, 1, … (modulo 2^k)
  - A is allowed to transmit up to W frames without waiting for an ACK
  - B can receive up to W consecutive frames
  - ACK J (or RR J), where 0<=J<= 2^k-1, sent by B means B is have received frames up to frame J-1 and is ready to receive frame J
- Window size, W can be less or equal to 2^k-1

Sliding Window Protocol (2)

- Example of Sliding-Window-Protocol: k = 3 bits, W = 7

Observations:

- A may tx W = 7 frames (F0, F1, …, F6)
- After F0, F1, & F2 are tx-ed, window is shrunk (i.e. can not transmit except F3, F4, …, F6)
- When B sends RR3, A knows F0, F1 & F2 have been received and B is ready to receive F3
- Window is advanced to cover 7 frames (starting with F3 up to F1)
- A sends F3, F4, F5, & F6
- B responds with RR4 when F3 is received – A advances the window by one position to include F2



W = distance between first unacknowledged frame and last frame that can be sent

Sliding Window Protocol - Piggybacking

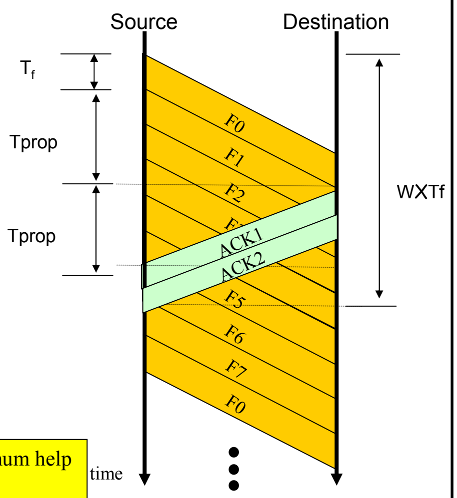
- When using sliding window protocol in full duplex connections:
  - Node A maintains its own transmit window
  - Node B maintains its own receive window
  - A frame contains: data field + ACK field
  - There is a sequence number for the data field, and a sequence number for the ACK field

Sliding Window Protocol - Efficiency

- Again we can distinguish two cases:
- Case 1: $W \geq 2a + 1$
- Case 2: $W < 2a + 1$

Sliding Window Protocol - Efficiency - Case 1

- Assume k=3, W = 7
  (ignoring Tack)
- Source can continuously keep transmitting!!
  - Because the ACK can arrive to source before the window is completed
- Utilization = 100%



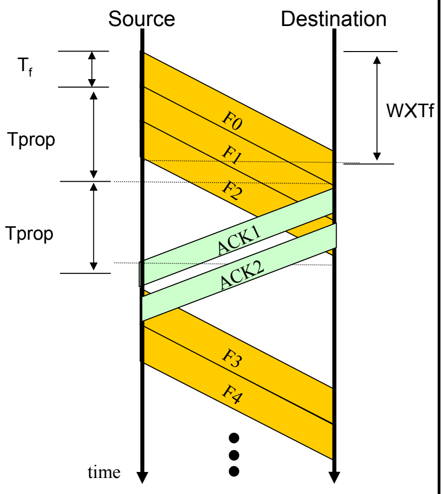
Sending ACK0 as soon as F0 is received is the maximum help the destination can do to increase utilization

---

Sliding Window Protocol - Efficiency - Case 2

- Assume k = 3, W = 3 (ignoring Tack)
- Source can NOT continuously keep transmitting!!
  - Because the ACK can NOT arrive to source before the window is completed
- Utilization = $\dfrac{W \times Tf}{Tf + 2 \times Tprop}$

$$= \frac{W}{1 + 2a}$$



Sliding Window Protocol - Efficiency

- Refer to Appendix A
- When window size is W (for error free), link utilization, U, is given by

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

where a = Tprop/Tf or length of link in bits

- Sliding window protocol can achieve 100% utilization if W >= (2a + 1)

Sliding Window Protocol

- Animation for Sliding Window protocol
- Sliding Window Protocol Simulation (http://www.cs.stir.ac.uk/~kjt/software/comms/jasper/SWP3.html)

Go-Back-N ARQ

- Based on the sliding-window flow control procedure - Sliding Window Protocol slide
- Three types of errors:
  1. i^{th} frame damaged:
     a. If A send subsequent frames (i+1, i+2, …), B responds with REJ i ➔ A must retransmit i^{th} frame and ***all subsequent frames***
     b. If A does not send subsequent frames and B does not respond with RR or REJ (since frame was damaged) ➔ timeout timer at A expires – send a POLL signal to B; B sends an RR i, i.e. it expect the i^{th} frame – A sends the i^{th} frame again
  2. Damaged RR (B receives i^{th} frame and sends RR i+1 which is lost or damaged):
     a. Since ACKs are cumulative – A may receive a subsequent RR j (j >i+1) before A times out
     b. If A times out, it sends a POLL signal to B – if B fails to respond (i.e. down) or its response is damaged subsequent POLLs are sent; procedure repeated certain number of time before link reset
  3. Damaged REJ – same as 1.b

Check for status of B before resending the frame

Go-Back-N ARQ – Efficiency With Errors

- Remember that Go-back-N ARQ utilization for error-free channels is given by:

  U = 1 for W > 2a + 1
  = W/(2a+1) for W < 2a + 1

- Assume a data frame can be in error with probability P
- With Go-back-N if one frame in error, we may retransmit a number of frames, on average K, and NOT only one!
- The average number of transmitted frames to transmit one frame correctly, Nr, is given by

  Nr = E[number of transmitted frames to successfully transmit one frame]
  = Σ f(i) x P^{i-1}(1-P)

- If a frame is transmitted i times (i.e. first (i-1) times are erroneous while it was received correctly in the i^{th} time), then f(i) is the total number of frame transmissions if our original frame is in error.
- f(i) is given by

  f(i) = 1 + (i-1)K

- Substituting f(i) in the above relation, yields

  Nr = (1-P+KP)/(1-P)

Identities:
Σ (X^{i-1},i=1,∞)=1/(1-X) for -1<X<1
Σ (iX^{i-1},i=1,∞) =1/(1-X^2) for -1<X<1

- Examining the operation of Go-back-N, an approximate value for K is 2a+1
- Then utilization with errors is given by

  U = (1-P)/(1+2aP) for W > 2a+1
  (1-P)W/{(2a+1)(2-P+WP)} for W < 2a+1

Again, expression reduces to the previous result if you set P = 0

Selective-Reject ARQ

- In contrast to Go-Back-N, the only frames retransmitted are those that receive –ve ACK (called SREJ) or those that time out
- More efficient:
  - Rx-er must have large enough buffer to save *post*-SREJ frames
  - Buffer manipulation – re-insertion of out-of-order frames

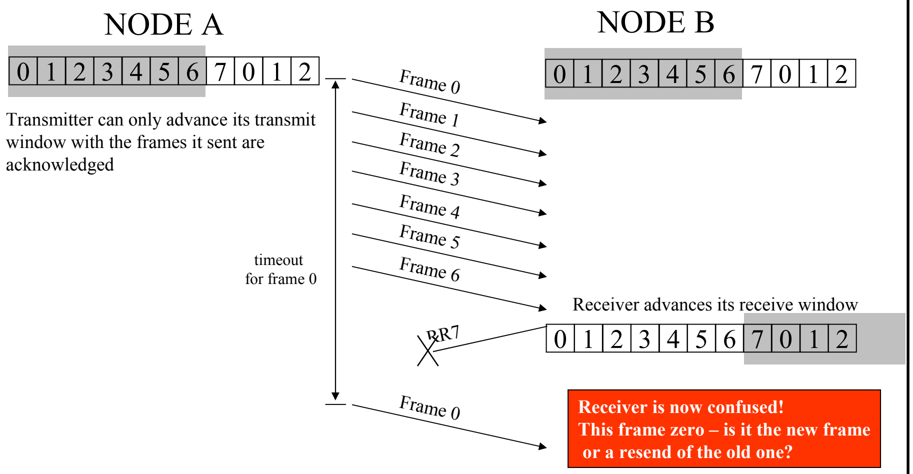
---

Window Size for Selective-Reject ARQ – Why?

- Window size: should less or equal to half range of sequence numbers
  - For n-bit sequence numbers, Window size is $\leq 2^{n-1}$ (remember sequence numbers range from 0,1, …, 2^n-1)
- Why? See next example

Window Size for Selective-Reject ARQ – Why? (2)

- Example: Consider 3-bit sequence number and window size of 7



NODE A

0 1 2 3 4 5 6 7 0 1 2

Transmitter can only advance its transmit window with the frames it sent are acknowledged

timeout
for frame 0

Frame 0
Frame 1
Frame 2
Frame 3
Frame 4
Frame 5
Frame 6

NODE B

0 1 2 3 4 5 6 7 0 1 2

Receiver advances its receive window

0 1 2 3 4 5 6 7 0 1 2

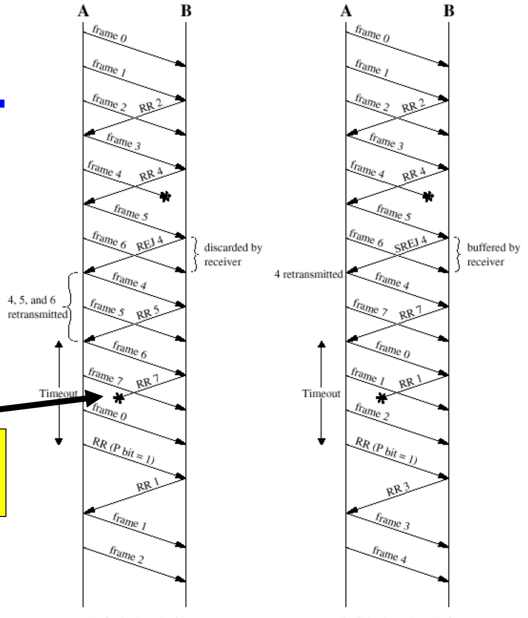
RR7

Frame 0

Receiver is now confused!
This frame zero – is it the new frame or a resend of the old one?

Go-Back-N/Selective-Reject ARQ Examples

- With Go-back-N frames 4,5 and 6 are retransmitted
- With Selective-Reject only frame 4 is retransmitted



(a) Go-back-N ARQ

(b) Selective-reject ARQ

Did this lost RR7 affect flow?
How did the link recover?

Selective Reject ARQ – Efficiency With Errors

- Remember that Selective Reject utilization for error-free channels is given by:

U = 1 for W > 2a + 1
= W/(2a+1) for W < 2a + 1

- Assume a data frame can be in error with probability P
- With Selective Reject if one frame in error, we retransmit only the required frame
- The average number of transmitted frames to transmit one frame correctly, Nr, is given by

Nr = E[number of transmitted frames to successfully transmit one frame]
= Σ i x $P^{i-1}(1-P)$ = 1/(1-P)

- Then utilization with errors is given by

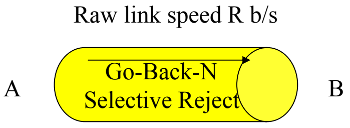
U = 1-P for W > 2a+1
(1-P)W/(2a+1) for W < 2a+1

Identities:
$\Sigma (X^{i-1}, i=1,\infty) = 1/(1-X)$ for $-1<X<1$
$\Sigma (iX^{i-1}, i=1,\infty) = 1/(1-X^2)$ for $-1<X<1$

Again, expression reduces to the previous result if you set P = 0

Important Performance Figures - Again

- Utilization (U) – fraction of time the link is used for transmitting data
- Throughput (b/s) – effective b/s as experienced by user data
  - Throughput = R * U (b/s)
- Throughput (frame/s) – average data frames per second the link is supporting
  - If U is equal to 100%
    ➔ Throughput = 1/Tf (frame/sec)
    = R*U/data_frame_size (frame/sec)
  - If U is LESS than 100%
    ➔ Throughput = W/T_totla (frame/sec)
    = R*U/data_frame_size (frame/sec)

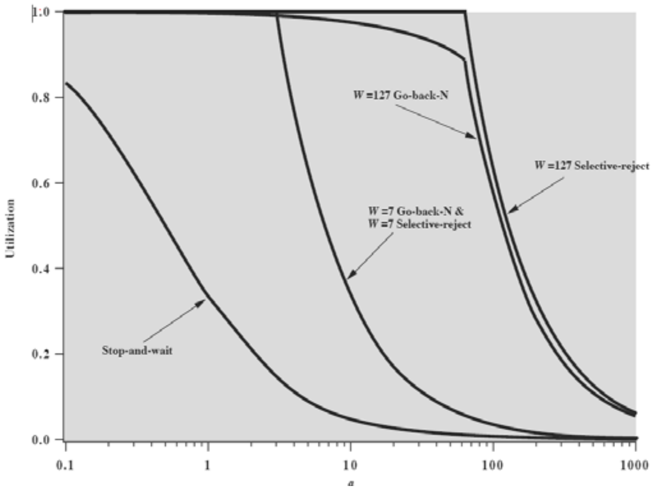


Utilization = ?
Throughput = ?

ARQ Utilization as a Function of a

- Remember a is given by Tprop/Tf – i.e. the length of the link in bits
- The curves are for P = 10^{-3}
- Note for W = 1, go-back-N and selective reject degenerate to the case of stop-and-wait

- Please note that the previous analyses are only approximate – errors in ACKs were ignored. Furthermore, in the case of go-back-N, errors in retransmitted frames other than the frame of interest were also ignored



Example:

Problem: Two neighboring nodes A and B use a sliding-window protocol with a 3-bit sequence numbers. As the ARQ mechanism, go-back-N is used with a window size of 4. Assuming A is transmitting and B is receiving, show the window positions for the following succession of events:

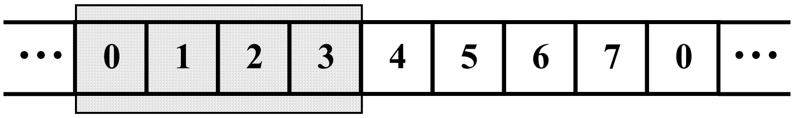
a) Before A sends any frames

b) After A sends frame 0, 1, 2 and B acknowledges 0, 1 and the ACKs are received by A

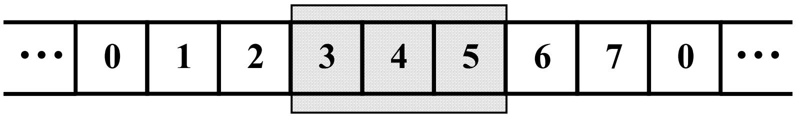
c) After A sends frames 3, 4, and 5 and B acknowledges 4 and the ACK is received by A

Example: Solution

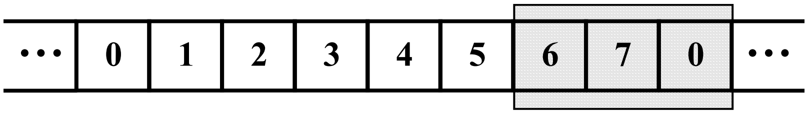
a)

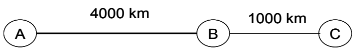


b)



c)





Example:

Problem: In the shown figure, frames are generated at node A and send to node C through node B. The following specifies the two communication links:

- **The data rate between node A and node B is 100 kb/s**
- **The propagation delay is 5 μsec/km for both links**
- **Both links are full-duplex**
- **All data frames are 1000 bits long; ACK frames are separate frames of negligible length**
- **Between A and B sliding window protocol with a window size of 3 is used**
- **Between B and C, stop-and-wait is used.**

There are no errors (lost or damaged frames)

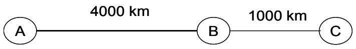
a) Calculate the utilization for link AB?

b) What is the throughput for link AB in bits per second? What is the throughput in frames per second?

c) Calculate the minimum rate required between nodes B and C so that the buffers of node B are not flooded.

d) What is the efficiency of the communication on link BC?

Example:



Solution:

Link AB: T_{f_AB} = frame length / R_{AB} = 1000/100 = 10 msec
T_{prop_AB} = 4000 km X 5 μsec = 20 msec
Link BC: T_{f_BC} = frame length / R_{BC} = 1000 / R_{BC}
T_{prop_BC} = 1000 km X 5 μsec = 5 msec

a) $a_{AB} = T_{prop\_AB} / T_{f\_AB}$ = 20 / 10 = 2
W = 3 is equal or less than (2 X a_{AB} + 1) = 5
➔ Utilization = W/(2 X a_{AB} + 1) = 3/5 = 60%

b) Throughput = 100 X 0.6 = 60 kb/s;
Throughput = 60 kb/s / (1000 bit) = 60 frame/second

c) Throughput for link BC in frames/second = 1 /(T_{f_BC} + 2XT_{prop_BC})
= 1/(1000/R_{BC} + 2X5X10^{-3})
= 1/(1000/R_{BC} + 10^{-2})

For not overflowing: frame throughput for link AB should be less or equal to frame throughput for link BC

➔ 60 <= 1/(1000/R_{BC} + 10^{-2})
1/60 >= 1000/ R_{BC} + 10^{-2}
1/60 - 10^{-2} >= 1000/ R_{BC}
R_{BC} >= 1000/(1/60 - 10^{-2}) = 150 kb/s

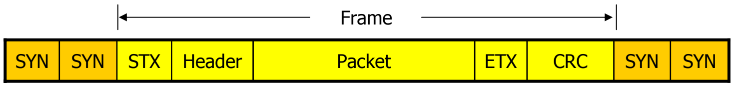
d) T_{f_BC} = 1000/150 kb/s = 6.667 msec
Efficiency (utilization) of link BC: $a_{BC} = T_{prop\_BC}/T_{f\_BC}$ = 5/6.666 = 0.75;
➔ Efficiency = 1/(2a+1) = 1/(2*0.75+1) = 40%

Framing

- How will the receiving DLC decide on the frame boundaries?
- How will the two ends of the DLC remain in sync?
- Three types of framing:
    - Character-based framing
    - Bit-oriented framing
    - Length counts

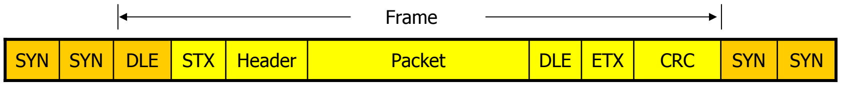
Character-Based Framing

- Utilizes character codes such as ASCII
- A 7-bits code ➔ 128 distinct codes
  - 96 printable characters (26 upper case letter, 26 lower case letters, 10 decimal digits, 34 non-alphanumeric characters)
  - 32 non-printable character
    - Formatting effectors (CR, BS, …)
    - Info separators (RS, FS, …)
    - Communication control (STX, ETX, …)
- A parity bit may be used
- Special characters:
  - SYN – synchronous idle – idle fill between frames when no data
  - STX – start of text
  - ETX – end of text



Character-Based Framing – cont’d

- What happens if an error produces control character in the header or CRC field?
- If the packet field is an “arbitrary bit string” – it too could contain character
  - ETX – leads to false frame boundary
  - Any other character
- Solution to false ETX – transparent mode
  - Insert DLE (data link escape) before STX character
  - Insert DLE before intentional use of communication control characters within the frame
- What is DLE itself appears in the binary data field
  - Insert another DLE (stuffing) for every appearance of DLE
  - Receiving DLC can strip off the first DLE from the arriving pair



Bit Oriented Framing

- Frame can be of any length – need not be multiples of 8
    - Subject to minimum and maximum
- A flag is used to identify the end of the frame
- Flag = a known bit string (similar to DLE ETX) that indicates the end of frame
- Bit stuffing – a process to prevent the occurrence of the flag in the data string
- The flag string is 01111110 or 01^60
    - 1^j notation means a string of j ones
- Bit stuffing:
    - Sender rule – insert a 0 after the appearance of *five* successive 1s
    - Receiver rule – the first 0 after each string of five consecutive 1's is deleted
        - If the five consecutive 1's are followed by 1 ➔ this is a flag

---

Bit Oriented Framing – cont'd

- Example of bit stuffing
    - A 0 is stuffed after each consecutive five 1's in the original frame
    - A flag, 01111110, without stuffing is sent at the end of the frame

					0							0					0							0
1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0
1	2	3	4	5	6		1	2	3	4	5	1	2	3	4	5			1	2	3	4	5	

1	1	1	1	1	0	1	0	1	1	1	1	1	0	1	1	1	1	1	0	1	0	1	1	1	1	1	0	0
1	2	3	4	5		6		1	2	3	4	5		1	2	3	4	5				1	2	3	4	5		

Other Uses of Bit Stuffing

- Abort capability of standard DLCs
- A frame can be aborted by sending 7 or more 1’s in a row
- A link is regarded idle if 15 or more 1’s arrive in a row
- Distinguishing normal vs. abnormal termination
  - A 01^6 followed by a 0 ➔ normal termination
  - A 01^6 followed by a 1 ➔ abnormal termination
- Bit stuffing guards against 01^6 pattern in data

- Breaking long sequence of 1’s
  - Converts to shorter sequences of 1’s
  - Useful for older modems to avoid loss of synchronization

Overhead Calculations for Bit-Oriented Framing

- Assume a frame of length K,
- Frame flag is 01^j for some j
  - i.e. 01^j0 is the flag, while 01^{j+1} is abnormal termination
- Assume all bits are iid with Prob[bit=0] = Prob[bit=1] = 1/2
- An insertion will occur at the i^{th} bit of the original frame (i>=j) if the string from i-j+1 to i is 01^{j-1}
  - The probability of this event is 2^{-j}
- An insertion will occur (for i>=2j-1) if the string from i-2j+2 to i is 01^{2j-2}
  - The probability of this event is 2^{-2j+1}
- The former term is ignored – also insertions due to yet longer string of 1’s are also ignored
- An insertion at position j-1 in the frame happens if the first j-1 bits are 1’s
  - The probability of this event happening is 2^{-j+1}

Overhead Calculations for Bit-Oriented Framing – cont'd

- The expected number of insertions is the sum of the expected insertions per position, i.e.

  $= 2^{-j+1} + 2^{-j}(K-j+1)$

  $= (K-j+3)2^{-j}$
- We have also j+1 bits for the end flag, the expected overhead is given by

  $E\{OV\} = (E\{K\} - j + 3)2^{-j} + j + 1$
- Since E{K} is typically >> j ➔

  $E\{OV\} <= E\{K\}2^{-j} + j + 1$
- Minimum overhead occurs at $j = \lfloor \log_2 E\{K\} \rfloor$
- Where minimum overhead is given by $E\{OV\} <= \log_2 E\{K\} + 2$

(K-j+1) positions

Example for j=3										
	0	2^{-j+1}	2^{-j}	2^{-j}	2^{-j}	.	.	.	2^{-j}	Probability of insertion
	1	2	3	4	.	.	.	.	K	Bit position
		j-1	i>=j							

Length Fields

- An alternative to end flags or special characters is to include a length field
- Length of the length field should be at least $\lfloor \log_2 K_{max} \rfloor + 1$
  - Kmax is the maximum frame length
  - Similar to the overhead for bit-oriented framing
- Could any other method of encoding frame lengths require smaller expected number of bits?
  - Information theory
  - Given any probability assignment P(K) on frame lengths, then the minimum expected number of bits that can encode such lengths is at least the entropy of that distribution, given by

    $H = \Sigma P(K) \log_2 P(K)^{-1}$
  - Example – let P(K) = 1/Kmax (i.e. all lengths are equally probable) ➔ $H = \log_2 Kmax$
  - Example – let $P(K) = p(1-q)^{K-1}$ where $p = 1/E\{K\}$ ➔ $H = \log_2 E\{K\} + \log_2 e$ for large E{K}

Source Coding for Frame Lengths

- The idea is to
  - map more likely values of K into short bit strings
  - Map less likely values of K into longer bit strings
- That is map a given K into $\log_2 P(K)^{-1}$ bits
- For geometric distribution of K ➔
  - Maximum # of required bits
  - The resulting code is called: Unary-binary encoding
- Unary-Binay Encoding:
  - For some j, K is written as $K = i2^j+r$ $(0<=r<2^j)$ – that is the number K is written in terms of i integer multiples of 2^j plus a remainder.
  - The encoding is then given by [Code_i,Code_r] – where Code_i is the binary representation for i while Code_r is the binary representation for r. Note Code_r is j bits wide, while Code_i can be of any size depending on i
- Example: K = 7 for j=2 ➔ $K = 1x2^2+3$ ➔ Unary-Binary Encoding is given by 111 – note Code_i is 1 while Code_r is 11.
- Example: K = 30 for j=2 ➔ $K = 7x2^2+2$ ➔ Unary-Binary Encoding is 11110 – note Code_i =111 while Code_r = 10
- Overhead: K is mapped into a bit string of length $\lfloor K/2^j \rfloor + 1+j$, then the expected overhead is given by $E\{OV\} = E\{K\}2^{-j}+1+j$
  - This is the same results of bit-oriented framing

Framing With Errors

- All previous framing techniques are sensitive to errors
- Bit-oriented framing with a flag is the least sensitive
  - If an error happens in a flag – another flag eventually appears and an erroneous packet is created
  - ARQ handles the problem
- Refer to textbook for partial solutions to the above problem used in DECNET; longer CRC; etc.

Maximum Frame Size

- Most networks accept variable frame sizes
- Large frame size
    - Transmission and processing efficiency
- Small frame size
    - Reduced frame errors
    - Real-time applications
    - Reduce network congestion/load
    - Pipelining effect (refer to next slide)
- Fixed frame size
    - Simplifies (speeds) network hardware
    - E.g. ATM – cells are 53 bytes

Pipelining is Packet Transmission

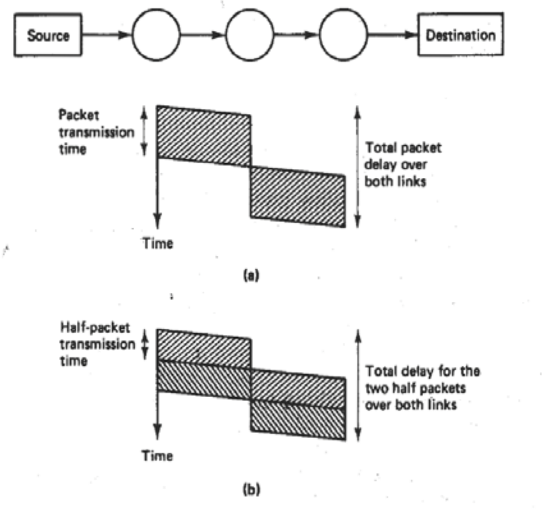


Figure 2.37 Decreasing delay by shortening packets to take advantage of pipelining. (a) The total packet delay over two empty links equals twice the packet transmission time on a link plus the overall propagation delay. (b) When each packet is split in two, a pipelining effect occurs. The total delay for the two half packets equals 1.5 times the original packet transmission time on a link plus the overall propagation delay.

Optimal Frame Size

- Assume:
  - Each frame contains a fixed no of bits, V, as overhead
  - Maximum length of packet = Kmax
  - Message length = M
- No of required frames = $\lceil M/Kmax \rceil$
  - Each of the first $\lceil M/Kmax \rceil$ are of length equal to Kmax
  - The last packet contains less than Kmax bits
- Total bits in frames

  total bits = $M + \lceil M/Kmax \rceil V$
- For very long M, the fraction of $V/(V+Kmax)$ is the overhead

Optimal Frame Size – with Pipelining

- Assume:
  - A packet must be received completely before a node can transmit it
  - Packets must be transmitted over j equal-capacity links – equal to C
  - No queuing at nodes (lightly loaded)
  - No errors on links
  - $M \geq Kmax$
  - Message length is uniformly distributed
- The total time, T, is the time for first packet to travel over the first j-1 links plus the time it takes the entire message to travel over the last link
- The number of message bit transmission times, TC is given by

  $TC = (Kmax + V)(j-1) + M + \lceil M/Kmax \rceil V$
- Taking the expectation

  $E\{TC\} \approx (Kmax + V)(j-1)+E\{M\}+E\{M\}V/Kmax + V/2$
- Optimal packet size, Kmax, that minimizes TC is given by

  $Kmax \approx \sqrt{(E\{M\}V/(j-1))}$
- Trade-offs:
  - For large V, optimal Kmax should be large
  - For large number of links, Kmax should be smalls

High-Level Data Link Control Protocol (HDLC)

- One of the most important data link control protocols
- Basic Characteristics:
    - Primary Station: issues *commands*
    - Secondary Station: issues *responses* – operates under the control of a primary station
    - Combined Station: issues commands and responses
- Two link configurations are defined:
    - Unbalanced: one primary plus one or more secondary
    - Balanced: two combined (functions as primary and/or secondary) stations

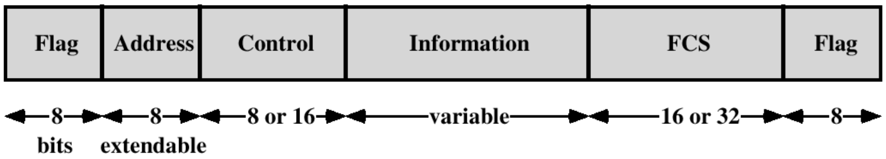
High-Level Data Link Control Protocol (HDLC) (2)

- Three transfer modes are defined:
    - Normal Response Mode (NRM) – used in unbalanced conf.; secondary may only tx data in response to a command from primary
    - Asynchronous Balanced Mode (ABM) – used in balanced conf.; either combined station may tx data without receiving permission from other station
    - Asynchronous Response Mode (ARM) – used in unbalanced conf.; Secondary may initiate data tx without explicit permission; primary still retains line control (initialization, error recovery, …)
- Animation for HDLC

HDLC - Applications

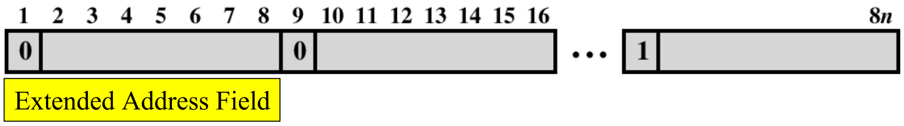
- NRM:
  - Point-multi-point (multi-drop line): one computer (primary) polls multiple terminals (secondary stations)
  - Point-to-point: computer and a peripheral
- ABM: most widely used (no polling involved)
  - Full duplex point-to-point
- ARM: rarely used

HDLC – Frame Structure – Flag Field



- Flag Field: unique pattern 01111110
  - Used for synchronization
  - To prevent this pattern form occurring in data ➔ *bit stuffing*
    - Tx-er inserts a 0 after each 5 1s
    - Rx-er, after detecting flag, monitors incoming bits – when a pattern of 5 1s appears; the 6th/7th bit are checked:
      - If 0, it is deleted
      - If 10, this is a flag
      - If 11, this is an ABORT
- Pitfalls of bit stuffing: one bit errors can split one frame into two or merge two frames into one

HDLC – Frame Structure - Address Field



- Address field identifies the secondary station that transmitted or is to receive frame
- Not used (but included for uniformity) for point-to-point links
- Extendable – by prior arrangement
- Address = 11111111 (single octet) used for broadcasting; i.e. received by all secondary stations

---

HDLC – Frame Structure - Control Field

	1	2	3	4	5	6	7	8
I: Information	0	N(S)			P/F	N(R)		
S: Supervisory	1	0	S		P/F	N(R)		
U: Unnumbered	1	1	M		P/F	M		

N(S) = Send sequence number
N(R) = Receive sequence number
S = Supervisory function bits
M = Unnumbered function bits
P/F = Poll/final bit

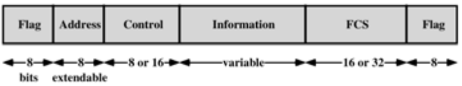
- First 2 bits of field determine the type of frame
  - Information frame (I): carry user data (upper layers) – flow and error control info is piggybacked on these frames as well
  - Supervisory frame (S): carry flow and error control info when there is no user data to tx
  - Unnumbered frame (U): provide supplementary link control
- Poll/Final (P/F) bit:
  - In command frames (P): used to solicit response from peer entity
  - In response frames (F): indicate response is the result of soliciting command

HDLC – Frame Structure - Control Field (2)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Information	0	N(S)							P/F	N(R)						
Supervisory	1	0	S		0	0	0	0	P/F	N(R)						

- "Set-mode" command ➔ extends control field to 16 bit for S and I frames
- Extension: 7-bit sequence numbers rather than 3-bit ones
- Unnumbered frames always use 3-bit sequence numbers

HDLC – Frame Structure – Information/FCS Fields



- Information field:
  - Present ONLY in I-frames and some U-frames
  - Contains integer number of octets
  - Length is variable – up to some system defined maximum
- FCS field:
  - Error detecting code
  - Calculated from *ALL* remaining bits in frame
  - Normally 16 bits (CRC-CCITT polynomial = $X^{16}+X^{12}+X^5+1$)
  - 32-bit optional FCS

HDLC Operation

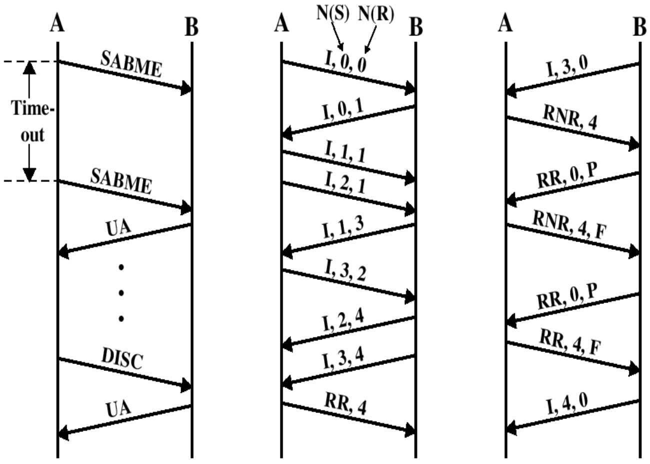
- Initialization
  - One side signals to the other the need for initialization
  - Specifies which of the three modes to use: NRM, ABM, or ARM
  - Specifies 3- or 7-bit sequence numbers
  - The other side can accept by sending unnumbered acknowledgment (UA)
  - The other side can reject by sending - A disconnected mode (DM) frame is sent
- Data Transfer
  - Exchange of I-frames: data and can perform flow/error control
  - S-frames can be used as well: RR, RNR, REJ, or SREJ
- Disconnect
  - DISC frame ➔ UA

HDLC – Operation

a) Link Setup & Disconnect:
  - SABM command – starts timer
  - B responds with UA (or DM if not interested)
  - A receives UA and initializes its variables
  - To disconnect: issue DISC command

b) Two-Way Data Exchange:
  - Full-duplex exchange of I-frames

c) Busy Condition:
  - Note the use of the P and F bits



(a) Link setup and disconnect (b) Two-way data exchange (c) Busy condition

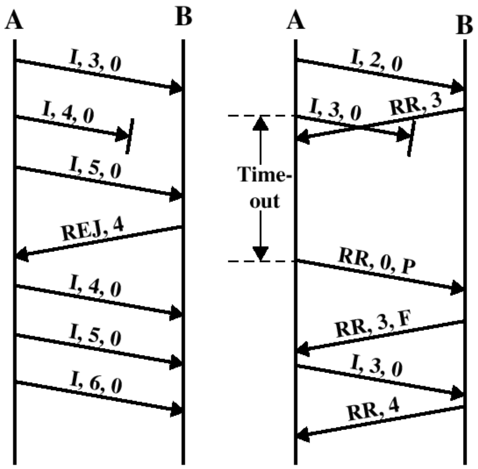
HDLC – Operation (2)

a) Reject Recovery:

- I-frame 4 was lost
- B receives I-frame 5 (out of order) – responds with REJ 4
- A resend I-frame 4 and all subsequent frames (Go-back-N)

b) Timeout Recovery:

- A sends I-frame 3 – but it is lost
- Timer expires before acknowledgement arrives
- A polls Node B
- B responds indicating it is still waiting for frame 3 – B set the F bit because this a response to A's solicitation



(d) Reject recovery (e) Timeout recovery

Other Data Link Control Protocols

- Link Access Procedure – Balanced (LAPB):
  - Part of X.25 packet-switching interface standard
  - Subset of HDLC – only ABM is provided
  - Designed for point-to-point
  - Frame format is same as HDLC
- Link Access Procedure – D-Channel (LAPD):
  - Part of ISDN – functions on the D-channel
  - 7-bit sequence numbers only
  - FCS field is always 16-bit
  - 16-bit address fields (two sub-addresses)

Other Data Link Control Protocols (2)

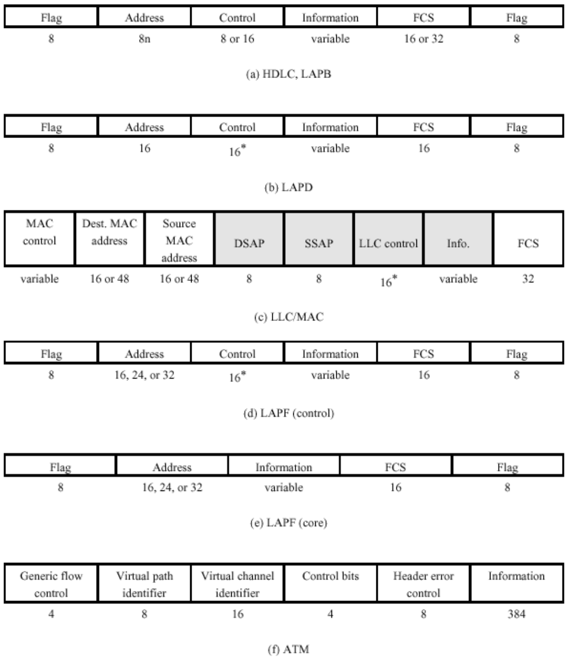
- Logical Link Control (LLC):
  - Part of IEEE802 family for LANs
  - Different frame format than HDLC
- Link Access Control Protocol for Frame-Mode Bearer Service (LAPF):
  - Designed for Frame Relay Protocol
  - Provides only ABM mode
  - Only 7-bit sequence numbers
  - Only 16-bit CRC field
  - Address field is 16, 24, or 32 bits long – containing a 10-bit, 16-bit, or 23-bit data link connection identifier (DLCI)
  - No control field – I.e. CANNOT do flow or error control (remember that frame relay was designed for fast and reliable connections!)

Other Data Link Control Protocols (3)

- Asynchronous Transfer Mode (ATM):
  - Like frame relay designed for fast and reliable links
  - NOT based on HDLC
  - New frame format – called CELL (53 bytes: 48 Bytes for payload or user data and 5 Bytes for overhead)
  - Cell has minimal overhead
  - NO error control for payload

Other Data Link Control Protocols (4)

- Frame Formats



Flag	Address	Control	Information	FCS	Flag
8	8n	8 or 16	variable	16 or 32	8

(a) HDLC, LAPB

Flag	Address	Control	Information	FCS	Flag
8	16	16*	variable	16	8

(b) LAPD

MAC control	Dest. MAC address	Source MAC address	DSAP	SSAP	LLC control	Info.	FCS
variable	16 or 48	16 or 48	8	8	16*	variable	32

(c) LLC/MAC

Flag	Address	Control	Information	FCS	Flag
8	16, 24, or 32	16*	variable	16	8

(d) LAPF (control)

Flag	Address	Information	FCS	Flag
8	16, 24, or 32	variable	16	8

(e) LAPF (core)

Generic flow control	Virtual path identifier	Virtual channel identifier	Control bits	Header error control	Information
4	8	16	4	8	384

(f) ATM

Point-2-Point Protocol at the Network Layer

- **Network layer main functions:**
  - **routing and flow control**
- **Other functions involving pairs of nodes**
  - **Transfer of packets between adjacent nodes or sites**
  - **You need to distinguish packets of one session from another**
- **The following material describes addressing and session identification**

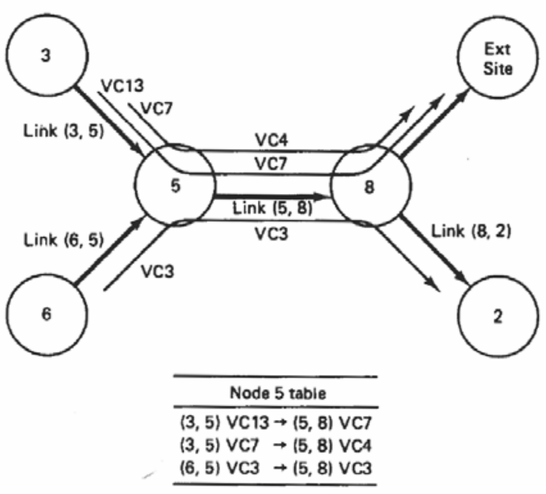
Session Identification and Addressing

- A received packet by a router must contain information to allow correct forwarding
  - Solution: let packets contain explicit addresses for source and destination sites and additional identification numbers to indicate the session within each site
  - Very general, but
  - Problem: lots of overhead
- Use virtual circuits
  - A virtual circuit identifies a path (a way) through the network at a given time
- A link may carry more than one virtual circuit (or sessions)
- Use of virtual circuit identifier
  - Usually encoded in binary
  - There is a maximum number of VCs per link
- Other methods of identification exist – to be discussed later

---

Session Identification and Addressing - Example

- Path for a session: 3→5→8
  - Uses virtual channel # 13 on link (3, 5)
  - Uses virtual channel # 7 on line (5, 8)
  - Node 8 will map VC # 7 to the external access link → destination
- Each link has its own set of virtual channels
  - No coordination needed to assign a VC
  - Each packet needs to contain virtual channel ID



Node 5 table

(3, 5) VC13 → (5, 8) VC7
(3, 5) VC7 → (5, 8) VC4
(6, 5) VC3 → (5, 8) VC3

Does a network utilizing VC have ordered delivery always? How?

Packet Numbering

- Datagram network – Problems
  - Packets may arrive out of order
  - Some packets may be lost
  - Some packets may be arbitrarily delayed
- Solution – use packet sequence numbers – modulo 2^k numbers
  - k-bit sequence number placed in the packet header – if the network layer at the source and destination have the responsibility of re-ordering and retransmission
  - In IP networks – the sequence number is placed in the transport layer header since reordering and retransmission is done at the transport layer
- Error events helped by sequence numbers:
  - Channel errors that lead to a frame that still satisfied CRC
  - If some nodes do not check CRC at DLC
  - If a link on the session fails → how many packets were successfully received? Which ones to retransmit?
  - If a node fails → stored packets are lost? Which ones to retransmit?
- ➔ Need error recovery at Network OR Transport layer

Error Recovery at Network or Transport Layer

- Very similar to ARQ at the DLC
  - Uses modulo m = 2k at the source site as SN
  - Destination site sends ACK containing RN equal (mod m) to the lowest-numbered yet-unreceived packet in the session
  - Can be go-back-N or selective-rejected
- Some differences exist
  - End-to-end recovery involving two sites (source and destination) and the subnet in between – For DLC two nodes with the link in between are involved only.
  - Sequence numbers: for end-to-end, the numbering is done per session; i.e. only packets or messages belonging to one session – For DLC all packets crossing the link are numbered sequentially.
  - Order: packets of one session may be out of order for network layer for example – For DLC all transmissions are in order and will be ordered at the other end of the link

Error Recovery at Network or Transport Layer – cont'd

- At Transport Layer check sum is used
  - Supplementary to CRC
  - E.g. TCP uses 16-bit long – 1's complement of the 1's complement sum of the 16 bit words making up the packet
  - Easy to compute in software
- At Network Layer
  - Again check sum could also be used
  - Since parts of the header do change (i.e. VC numbers), compute check sum over data part only or include the source and destination addresses

---

Flow Control

- Is achieved by the ARQ schemes described earlier
- With a window of size n, at most n packets can be in transmit
- HDLC utilizes RNR frames to stop the transmitter
- Flow control and congestion
  - Allow the window size to adapt to network congestion (e.g. TCP window)
  - Delayed ACKs
    - Difficult to distinguish between an intentionally delayed ACK and that because of congestion
  - Permit – allow receiver to change the window with each ACK
    - Sends two numbers RN and j – meaning the transmitter can send packets from number RN to RN + j – 1 inclusive
    - Used in X.25 network

Should Error Recovery Be AT Transport Layer or Network Layer?

- The natural place is at the Transport layer
  - Examples: TCP and TP4 (ISO transport layer)
- Larger number of interconnected networks that DO NOT provide reliable service
- Major disadvantage – transport can not distinguish between ACKs that are slow because of network congestion and those that will never arrive (may be lost)
  - Serious problem for TCP over wireless links
- Slowed or lost acks lead to more retransmissions → more congestion → slower and more losses → more congestion …
- The key is to make the networks more reliable (somehow) and ask little error-recovery of transport!

The X.25 Network Layer Standard

- Developed by CCITT to provide standard interface between external sites and subnet nodes
- Utilizes X.21 as physical layer and LAPB (a variant of HDLC) as DLC
- Packet structure (as shown) – significant of Q, D, M, and C bits
- Utilizes end-to-end VC
  - Designed for low 64kb/s links
  - Error control and flow control are provided both at data link and network layers
- Provides per-hop control
  - Flow control – using "permits"
  - Error control – using LAPB
- Virtual circuit establishments
  - Call-request / call-accept packets
  - Session setup
- Can be used to encapsulate IP datagrams!

1	2	3	4	5	6	7	8	
Q	D	Modulus		Virtual channel				Byte 1
Virtual channel								Byte 2
RN			M	SN			C	Byte 3

The Internet Protocol

- See material on 2.8.4
- Figure 2.51

The Transport Layer

- Main functions
- Two examples
    - TCP/IP suite
    - ISO – classes of TPs: class0, class1, …, class4
- The TCP layer will be covered in details later in the course

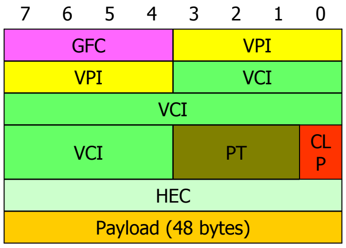
Asynchronous Transfer Mode (ATM)

- **1990's vision: Broadband Integrated Digital Services (B-ISDN)**
  - **ATM is the network layer**
  - **Provides speeds from 155-622 Mb/s**
  - **A slimmer version, ISDN, existed**
- **Provides INTEGRATED end-to-end voice, video and data communications on one network**
- **Main characteristics:**
  - **Provides degrees of QoS requirement – not available in IP**
  - **Facilitates digital telephony – 64kb/s circuits**
  - **Utilizes cell (packet) switching – short 53 bytes long packets**
  - **Can provide very high transport speeds for applications requiring extreme bandwidths such as high-resolution image/video, high-speed LAN connections, etc.**

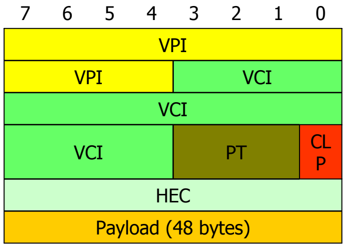
ATM Characteristics

- **STM versus ATM**
  - **ATM can provide STM-like services through its adaptation layer**
  - **See textbook discussion on pages 130 and 131**
- **ATM is suitable for bursty traffic**
- **Intended to operate over optical fibers – i.e. reliable links**
- **Cell structure**
  - **Header – 5 bytes – fixed length/structure to allow high speed switching**
  - **Payload – 48 bytes**
  - **Error detection/correction for header only**
  - **Note GFC is only present at the user-network interface (UNI)!**

ATM Cell at UNI



ATM Cell at NNI

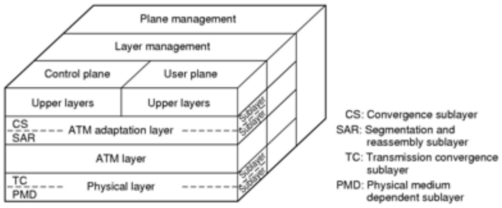


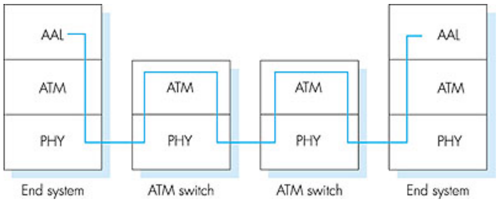
ATM Characteristics – cont'd

- No link-by-link retransmissions
  - Error control is only for the header part
  - No DLC functionality
  - Error recovery is on an end-to-end basis only
- SONET is one physical layer for ATM (STS-3 rate)
- Service Model:
  - Constant bit rate (CBR)
  - Variable bit rate (VBR)
  - Available bit rate (ABR)
  - Unspecified bit rate (UBR)
- Provides connection-oriented service with the aid of its adaptation layers
  - Cells arrive in order
  - reliability

ATM Protocol Stack

- Physical layer
- ATM layer – performs cell switching and routing
- ATM Adaptation layer (AAL)
  - Similar to transport layer in TCP/IP stack
  - Different kinds of AALs for different services
- AALs exist at the edge of the ATM network – performs segmentation and reassembly





ATM Adaptation Layer

- Adapts user traffic to ATM layer
- Provides four classes of services:
  - Class 1: Constant bit-rate traffic
  - Class 2: Variable bit-rate packetized data with fixed delay
  - Class 3: Connection oriented data
  - Class 4: Connectionless data
  - Class 5: IP datagrams

Class 3 – Connection Oriented Traffic

- To facilitate establishing connection oriented services
  - Reordering – Segment type bits and sequence numbers
  - Trailer containing length and CRC

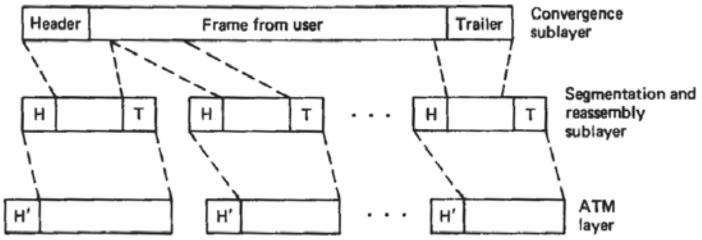


Figure 2.57 Headers and trailers at the convergence sublayer and segmentation and reassembly sublayer of the adaptation layer. Note that the user frame and its convergence sublayer header and trailer need not completely fill the data area at the segmentation and reassembly sublayer. The data plus header and trailer at the segmentation and reassembly sublayer must be exactly 48 bytes in length, however, to fit into the ATM data area.

Class 4 – Connectionless Traffic

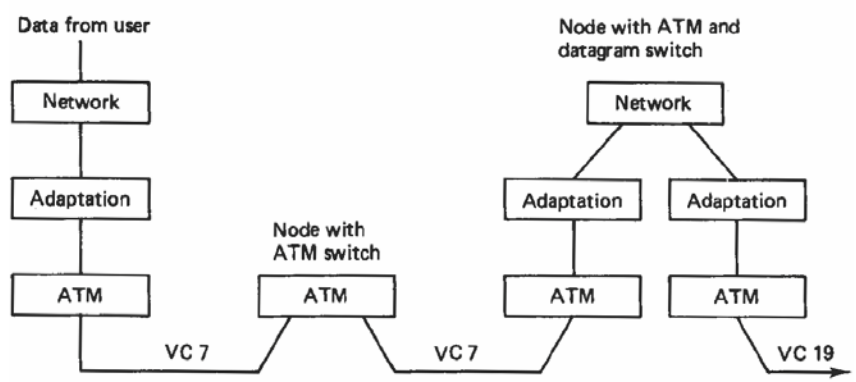


Figure 2.58 Datagram traffic can use an ATM network by being routed from one datagram switch to another. The virtual link between neighboring datagram switches is a virtual circuit in the ATM network dedicated to the aggregate of all the datagram traffic traveling between the two datagram switches.

Congestion Control

- Three mechanism supported
  - User and network agree on required bit rate at call setup
    - Mainly for class 1
    - Hard (not used) for class 2 and class 3
    - Can not be used for class 4
  - Adaptation layer does traffic policing and shaping – monitors user traffic
    - E.g. leaky bucket method
    - Excess traffic may be discarded or marked as low priority traffic
  - The priority bit (CLP) in the ATM cell header