Chapter 2: The Data Link Layer

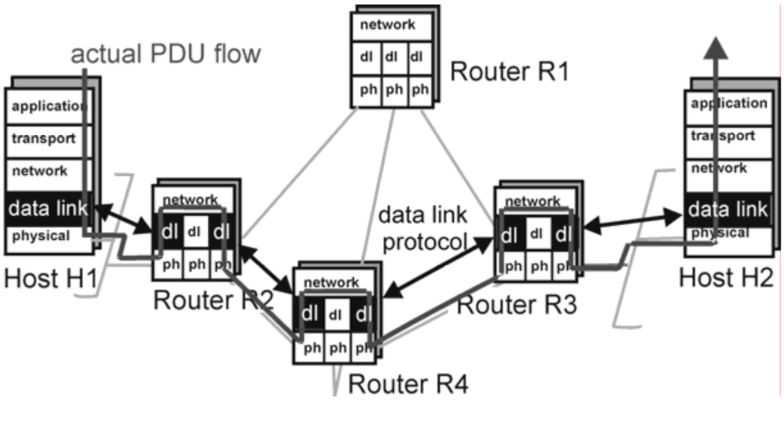
Our goals:

- understand principles behind data link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - reliable data transfer, flow control
- instantiation and implementation of various link layer technologies

Overview:

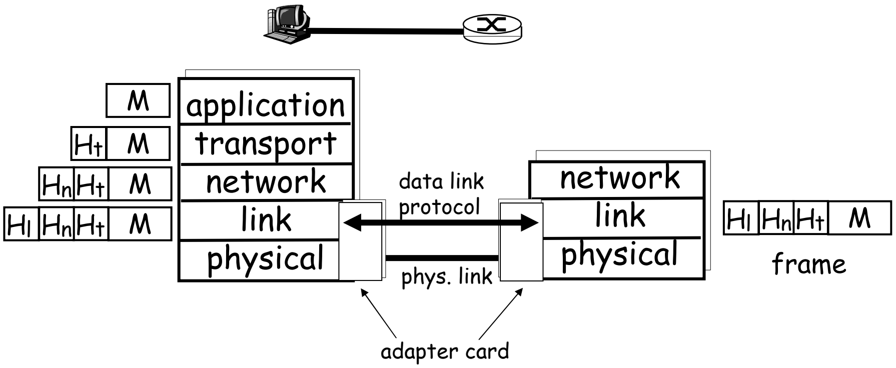
- link layer services
- error detection, correction
- specific link layer technologies:
  - PPP
  - ATM

Link Layer: setting the context



Link Layer: setting the context

- two *physically connected* devices:
  - host-router, router-router, host-host
- unit of data: *frame*



Link Layer Services

- Framing, link access:
  - encapsulate datagram into frame, adding header, trailer
  - implement channel access if shared medium,
  - 'physical addresses' used in frame headers to identify source, dest
    - different from IP address!
- Reliable delivery between two physically connected devices:
  - seldom used on low bit error link (fiber, some twisted pair)
  - wireless links: high error rates
    - Q: why both link-level and end-end reliability?

Link Layer Services (more)

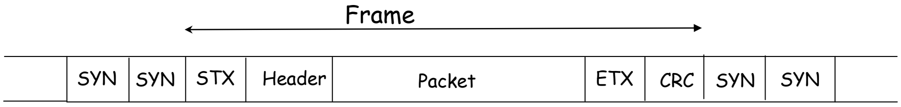
- Flow Control:
  - pacing between sender and receivers
- *Error Detection*:
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame
- Error Correction:
  - receiver identifies *and corrects* bit error(s) without resorting to retransmission

Framing

010100111010100100101010100111000100

- Where is the DATA??
  - Synchronous bit pipe (*idle fill*)
  - Intermittent synchronous bit pipe
  - Asynchronous bit pipe
- Three approaches to find frame and idle fill boundaries:
  - Character oriented framing
  - Length counts
    - - fixed length
  - Bit oriented protocols (flags)

Character-oriented framing



Frame

SYN	SYN	STX	Header	Packet	ETX	CRC	SYN	SYN	

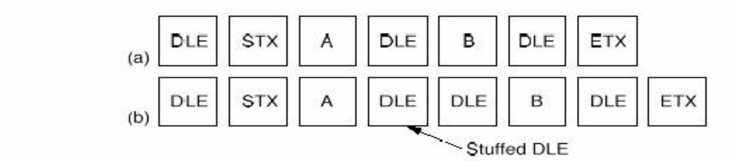
SYN is synchronous idle
STX is start text
ETX is end text

Standard character codes such as ASCII and EBCDIC contain **<u>special communication characters</u>** **that cannot appear in data**

- **Entire transmission is based on a character code**

Issues With Character-Based Framing

- Character code dependent
  - –How do you send binary data?
  - *Transparent mode*
    - *Use a special character (DLE)*
    - *Inserted before STX*
    - *Inserted before control character within a frame*
  - *What if DLE appears itself in the data?*



(a)	DLE	STX	A	DLE	B	DLE	ETX	
(b)	DLE	STX	A	DLE	DLE	B	DLE	ETX

  - Errors in control characters are messy
  - Frames must be integer number of characters

Bit-Oriented Framing: Flags

- Frame can have any length (subject to min. and max.)
- A flag is some fixed string of bits to indicate the start and end of a packet
  - A single flag can be used to indicate both the start and the end of a packet
- In principle, any string could be used, but appearance of flag must be prevented somehow in data
  - Standard protocols use the 8-bit string 01111110 as a flag
  - Use 01111111..1110 (<16 bits) as abort under error conditions
- Constant flags or 1's is considered an idle state
- Thus 0111111 is the actual bit string that must not appear in data
- The size of frame is data dependent!
- INVENTED ~ 1970 by IBM for SDLC (synchronous data link protocol)

Start and End Flags With Bit Stuffing

- In practice,
  - flag is a bit string 01^j0
  - 01^j followed by 1 indicates abnormal frame termination
    - Why?
    - The receiver discards the frame and hast to wait for next 01^j0
- Example: 01111110
    - Every time you see five 1's in sequence in the data, stuff a zero into the stream

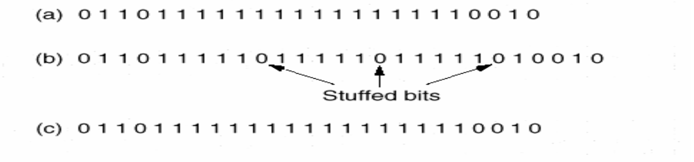

(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0
(b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0
Stuffed bits
(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Fig. 3-5. Bit stuffing. (a) The original data. (b) The data as they appear on the line. (c) The data as they are stored in the receiver's memory after destuffing.

- **Why is it necessary to stuff a 0 in 0111110?**

Performance Figures

- Expected overhead

$$E(OV) \le E(K)2^{-j} + j + 1$$

- Smallest j for minimum OV

$$j = \lfloor \log_2 E(K) \rfloor$$

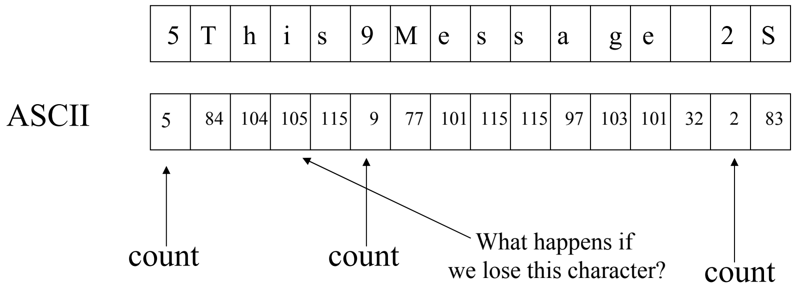
- $E(OV) \le \log_2 E(K) + 2$

Length field approach (DECNET)

- Simple alternative to flags or special character
- Use a header field to give the length of the frame (in bits or bytes)
  - Receiver can count until the end of the frame to find the start of the next frame
  - Receiver looks at the respective length field in the next packet header to find that packet's length
- Length field must be at least $\log_2 K \text{max} + 1$ bits long
  - This restricts the packet size to be used
- Issues with length counts
  - Difficult to recover from errors
  - Resynchronization is needed after an error in the length count

Character Count

Frames are prefixed by chars/frame



	5	T	h	i	s	9	M	e	s	s	a	g	e		2	S
ASCII	5	84	104	105	115	9	77	101	115	115	97	103	101	32	2	83

count

count

What happens if we lose this character?

count

Big Problem:

- How do we recover if we get out of sync?

Clock-Based Framing

- Used by SONET
- Fixed size frames (810 bytes)
- Look for start of frame marker that appears every 810 bytes
- Will eventually sync up

Final Framing Method - Physical Layer Coding Violations

- Start/End flag consists of sequence that is illegal in the data
- Example:
  - 10 is 1
  - 01 is 0
  - 00 or 11 could be used as flags

Framing with Errors

- All framing techniques are sensitive to errors
  - An error in a length count field causes the frame to be terminated at the wrong point (and makes it tricky to find the beginning of the next frame)
  - An error in DLE, STX, or ETX causes the same problems
  - An error in a flag, or a flag created by an error causes a frame to disappear or an extra frame to appear
- Flag approach is least sensitive to errors because a flag will eventually appear again to indicate the end of a next packet
  - Only thing that happens is that an erroneous packet was created
  - This erroneous packet can be removed through an error detection technique

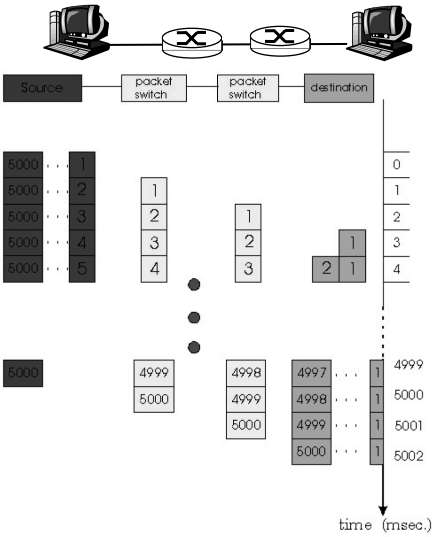
Partial solutions

- CRC for the header
- Put the length field into the trailer of the preceding frame
- Using longer CRC (32 bit)

Maximum Frame Size

- Variable frame length
  - Most existing networks use variable frame length
  - Large frame size
    - Transmission and processing overhead argues for large frame lengths
  - Small frame size
    - Pipelining effect
    - Heavy-load networks
    - Real-time applications
    - Transmission error
- Fixed frame length
  - To simplify the hardware for high-speed switching

Pipelining



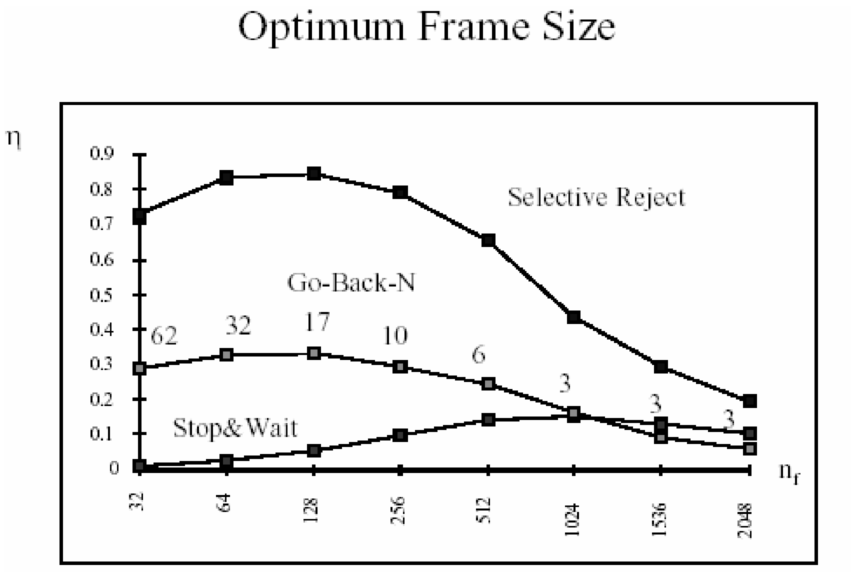
Simple analysis for Optimal frame size

$$TC = (K_{\max} + V)(j-1) + M + \left\lceil \frac{M}{K_{\max}} \right\rceil V$$

$$E(TC) \approx (K_{\max} + V)(j-1) + E(M) + \frac{E(M)V}{K_{\max}} + \frac{V}{2}$$

differentiate w/r to Kmax

$$K_{\max} \approx \sqrt{\frac{E(M)V}{j-1}}$$



Source: Communication Networks by Leon-Garcia 1998

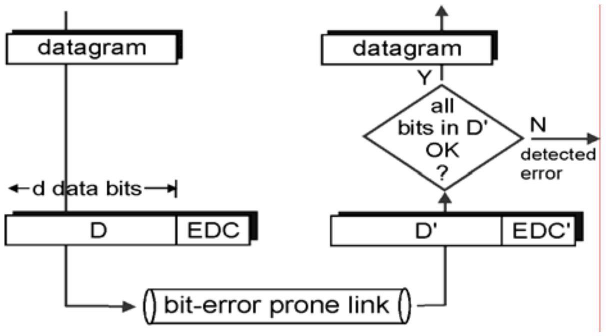
Fixed Length Packets (e.g., ATM)

- All packets are of the same size
  - In ATM networks all packets are 53 Bytes
- Requires synchronization upon initialization
- Issues:
  - Message lengths are not multiples of packet size
  - Last packet of a message must contain idle fill (efficiency)
  - Synchronization issues
  - Fragmentation and re-assembly is complicated at high rates

Error Detection

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

- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction



Error detection techniques

- Used by the receiver to determine if a packet contains errors
- If a packet is found to contain errors the receiver requests the transmitter to re-send the packet
- Error detection techniques
  - Parity check
    - single bit
    - Horizontal and vertical redundancy check
  - Cyclic redundancy check (CRC)

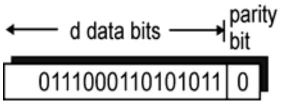
Effectiveness of error detection technique

- Effectiveness of a code for error detection is usually measured by three parameters:
  - minimum distance of code (d) (min # bit errors undetected)

    The minimum distance of a code is the smallest number of errors that can map one codeword onto another. If fewer than d errors occur they will always detected. Even more than d errors will often be detected (but not always!)
  - burst detecting ability (B) (max burst length always detected)
  - probability of random bit pattern mistaken as error free (good estimate if # errors in a frame >> d or B)
    - Useful when framing is lost
    - K info bits => 2^k valid codewords
    - With r check bits the probability that a random string of length k+r maps onto one of the valid 2^k codewords is $2^k / 2^{k+r} = 2^{-r}$

Single Bit Parity Checking

Detect single bit errors



Receiver counts number of ones in frame

– An even number of 1's is interpreted as no errors

– An odd number of 1's means that an error must have occurred

A single error (or an odd number of errors) can be detected

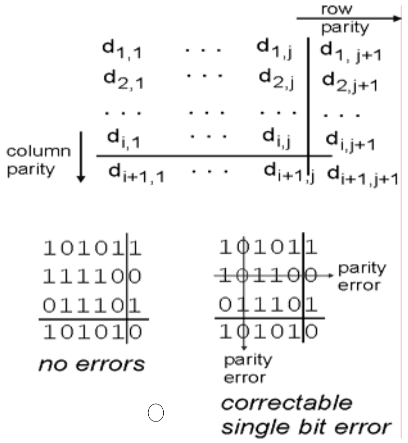
An even number of errors cannot be detected

Nothing can be corrected

- Probability of undetected error (independent errors) $\sum_{i\,even} \binom{N}{i} p^i q^{(N-i)}$

Two Dimensional Bit Parity Checking

Detect *and correct* single bit errors



row parity
$d_{1,1}$ . . . $d_{1,j}$ $d_{1,j+1}$
$d_{2,1}$ . . . $d_{2,j}$ $d_{2,j+1}$
. . . .
$d_{i,1}$ . . . $d_{i,j}$ $d_{i,j+1}$
column parity
$d_{i+1,1}$ . . . $d_{i+1,j}$ $d_{i+1,j+1}$
10101 1
11110 0
01110 1
10101 0
no errors
10101 1
10110 0
01110 1
10101 0
parity error
parity error
correctable single bit error

- The data is viewed as a rectangular array (i.e., a sequence of words)
- Even number of errors confined to a single row can be detected
- Minimum distance=4, any 4 errors in a rectangular configuration is undetectable

Internet checksum

Goal: detect "errors" (e.g., flipped bits) in transmitted segment (note: used at transport layer *only*)

Sender:

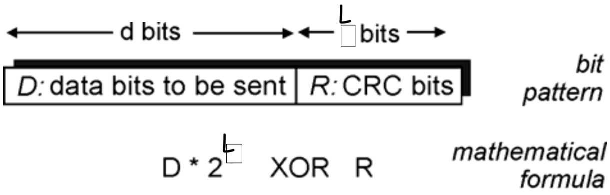
- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

Receiver:

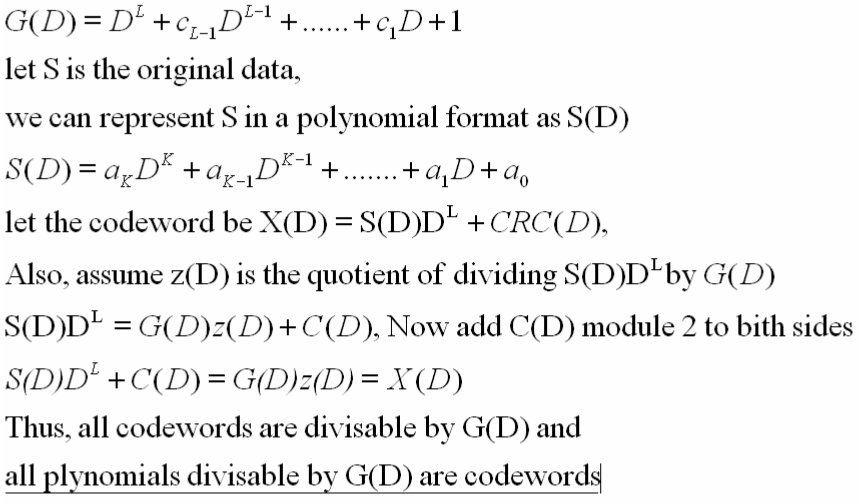
- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected. *But maybe errors nonetheless?* More later ….

Checksumming: Cyclic Redundancy Check

- view data bits, D, as a binary number
- choose L+1 bit pattern (generator), G
- goal: choose L CRC bits, R, such that
  - <D,L> exactly divisible by G (modulo 2)
  - receiver knows G, divides <D,L> by G. If non-zero remainder: error detected!
  - can detect all burst errors less than L+1 bits
- widely used in practice (ATM, HDCL)



Shift register circuit



$G(D) = D^L + c_{L-1}D^{L-1} + \ldots\ldots + c_1 D + 1$

let S is the original data,

we can represent S in a polynomial format as S(D)

$S(D) = a_K D^K + a_{K-1} D^{K-1} + \ldots\ldots. + a_1 D + a_0$

let the codeword be $X(D) = S(D)D^L + CRC(D)$,

Also, assume z(D) is the quotient of dividing $S(D)D^L$ by $G(D)$

$S(D)D^L = G(D)z(D) + C(D)$, Now add C(D) module 2 to bith sides

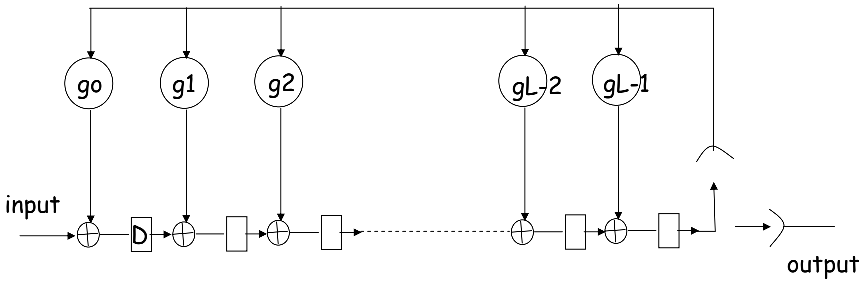
$S(D)D^L + C(D) = G(D)z(D) = X(D)$

Thus, all codewords are divisable by G(D) and

all plynomials divisable by G(D) are codewords

In Practice

- Can be implemented easily in hardware by the feedback shift register (e.g. VLSI)



CRC Example

Want:

$D \cdot 2^L$ XOR R = nG

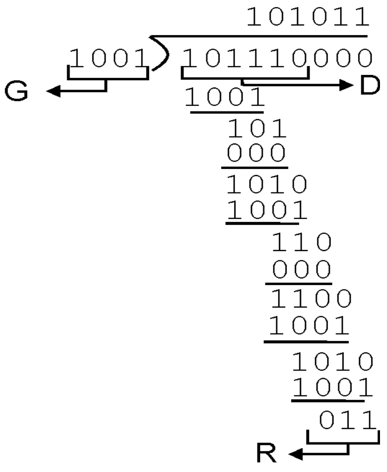
equivalently:

$D \cdot 2^L$ = nG XOR R

equivalently:

if we divide $D \cdot 2^L$ by G, want remainder R

$$R = \text{remainder}\left[\frac{D \cdot 2^L}{G}\right]$$



Performance of CRC

- For L check bits per frame and a frame length less than $2^L - 1$, the following can be detected
  - All patterns of 1,2, or 3 errors (d > 3)
  - All bursts of errors of L or fewer bits
  - Random large numbers of errors with prob. 1- 2^{-L}
- Standard DLC's use a CRC with L=16 with option of L=32
- CRC-16, G = $X^{16} + X^{15} + X^2 + 1$ =11000000000000101

Physical Layer Error Characteristics

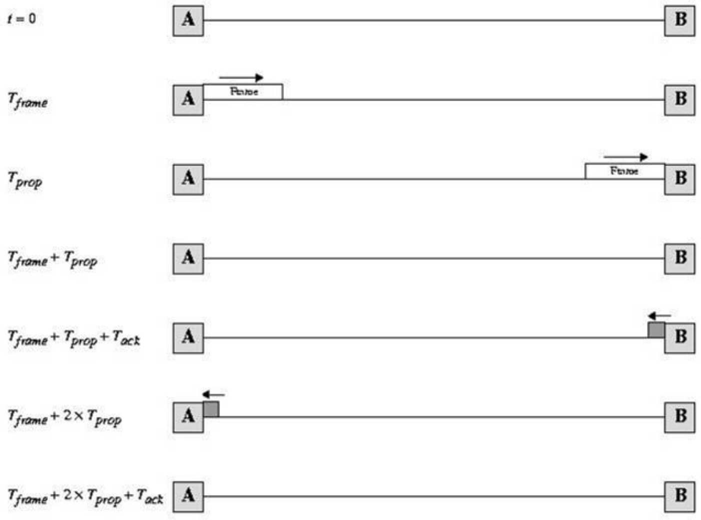
- Most Physical Layers (communications channels) are not well described by a simple BER parameter
- Most physical error processes tend to create a mix of random & bursts of errors
- A channel with a BER of 10^{-7} and a average burst size of 1000 bits is very different from one with independent random errors
  - Example: For an average frame length of 104 bits
    - – random channel: E[Frame error rate] ~ 10^{-3}
    - – burst channel: E[Frame error rate] ~ 10^{-6}
- Best to characterize a channel by its Frame Error Rate
- This is a difficult problem for real systems

ARQ: Retransmission Strategies

- Concept: to detect frames in error and then request the transmitter to repeat the erroneous frames
- Systems which automatically request the retransmission of missing packets or packets with errors are called ARQ systems.
- What about FEC (Forward Error Correction)?
- 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 Repeat

Stop-and-Wait ARQ

- The simplest form of flow/error control
- Operation:
  - a transmitting entity sends its PDU
  - an acknowledgment is sent by the receiving entity to indicates its willingness to receive another PDU
- 2 kinds of errors:
  - Damaged frame at destination
  - Damaged acknowledgement at source
- it works fine
- Major design issue:
  - the ratio between the link-bit length and PDU length



t = 0
A
B
T_{frame}
Frame
T_{prop}
$T_{frame} + T_{prop}$
$T_{frame} + T_{prop} + T_{ack}$
$T_{frame} + 2 \times T_{prop}$
$T_{frame} + 2 \times T_{prop} + T_{ack}$

Figure 11.4 Stop-and-Wait Link Utilization

Stop-and-wait ARQ varieties

- Pure Stop-and-Wait
  - Problem: lost packet
- Time-out Stop-and-Wait
  - Problem: receiver can't tell which packet
- Sequence number (module 2)
  - Instead of sending "ack" or "nak", the receiver sends the number of the packet currently awaited.
  - Sequence numbers and request numbers can be sent modulo 2.
  - This works correctly assuming that
    - Frames travel in order (FCFS) on links
    - The CRC never fails to detect errors
    - The system is correctly initialized.

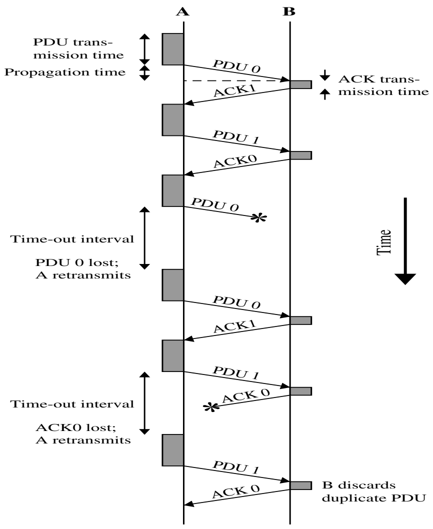


Figure 6.9 Stop-and-Wait ARQ

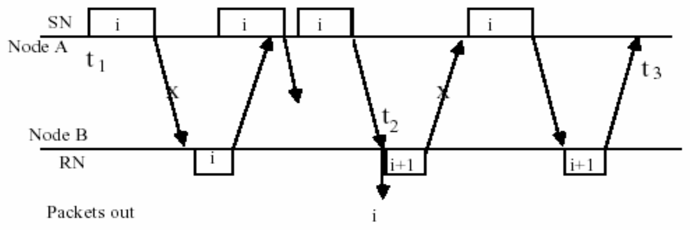
Correctness of stop & wait with integer SN, RN

- Assume, for A to (from) B transmission, that
  - All errors are detected as errors
  - Initially no frames are on link, SN=0, RN=0
  - Frames may be arbitrarily delayed or lost
  - Each frame is correctly received with at least some probability *q -> 0.*
- Split proof of correctness into two parts:
  - SAFETY: show that no packet is ever released out of order or more than once
  - LIVENESS: show that every packet is eventually released (i.e. no deadlock condition)

Safety

- No frames on link initially, packet 0 is first packet accepted at A, it is the only packet assigned SN=0, and must be the packet released by B if B ever releases a packet
- Subsequently (using induction) if B has released packets up to and including n-1, then RN is updated to n when n-1 is released, and only n can be released next

LIVENESS



- t1 = time at which A first starts to transmit packet i
- t2 = time at which B correctly receives & releases i, and increases RN to i+1
- t3 = time at which SN is increased to i+1 Will prove that t1 < t2 < t3 < ∞ => Liveness

Liveness Argument

- Let SN(t), RN(t) be values of SN and RN at time t
- **From the algorithm,**
  - SN(t) and RN(t) are increasing in t and SN(t) <= RN(t) for all t
  - From safety (since i has not been sent before t1) RN(t_1) <= i and SN(t_1) = i
- Therefore, RN(t_1) = SN(t_1) = i
- RN is incremented at t_2 and SN at t_3, so $t_2 < t_3$
- A transmits i repeatedly up to t3, and thus to t_2 when it is correctly received. Since *q > 0*, t_2 is finite
- B transmits RN=i+1 repeatedly until correctly received at t_3, and q>0 implies that t_3 is finite.

Correctness of Stop & Wait with binary (finite) SN, RN

- Assume that frames travel on link in order
  - Note that with integer SN, RN, either
  - SN=RN (from t_1 to t_2) or (3)
  - SN=RN-1 (from t_2 to t_3) (4)
- Since frames travel in order, the sequence numbers arriving at B and the request numbers arriving at A are increasing, so a single bit can resolve the ambiguity between (3) and (4)
- RN = 0 and SN = 1 or RN =1 and SN = 0

  => received packet is an old packet
- RN = 0 and SN = 0 or RN = 1 and SN = 1

  => received packet is new

Error-Free Stop and Wait

$T = T_{frame} + T_{prop} + T_{proc} + T_{ack} + T_{prop} + T_{proc}$

T_{frame} = time to transmit frame
T_{prop} = propagation time
T_{proc} = processing time at station
T_{ack} = time to transmit ack

Assume T_{proc} and T_{ack} relatively small

Error-Free Stop-and-Wait

$T \approx T_{frame} + 2T_{prop}$

Throughput = $1/T = 1/(T_{frame} + 2T_{prop})$ frames/sec

Normalize by link data rate: $1/T_{frame}$ frames/sec

$$U = \frac{1/(T_{frame} + 2T_{prop})}{1/T_{frame}} = \frac{T_{frame}}{T_{frame} + 2T_{prop}} = \frac{1}{1 + 2a}$$

where $a = T_{prop} / T_{frame}$

The Parameter a

$$a = \frac{\text{propagation time}}{\text{transmission time}} = \frac{T_{\mathrm{Prop}}}{T_{Frame}} = \frac{d/V}{L/R}$$

where

d = distance between stations
V = velocity of signal propagation
L = length of frame in bits
R = data rate on link in bits per sec

Rd/V ::= bit length of the link
a ::= ratio of link bit length to the length of frame

Stop-and-Wait Link Utilization

- If T_{prop} large relative to T_{frame} then throughput reduced
- If propagation delay is long relative to transmission time, line is mostly idle
- Problem is only one frame in transit at a time
- Stop-and-Wait rarely used because of inefficiency

Stop and wait in the presence of errors

- Let P = the probability of an error in the transmission of a packet or in its acknowledgment
- $T \approx T_{frame} + 2T_{prop}$
- X = the amount of time that it takes to transmit a packet and receive its ACK. This time accounts for retransmissions due to errors
- $E[X] = T/(1-P)$, Efficiency = $T_{frame} / E[X]$

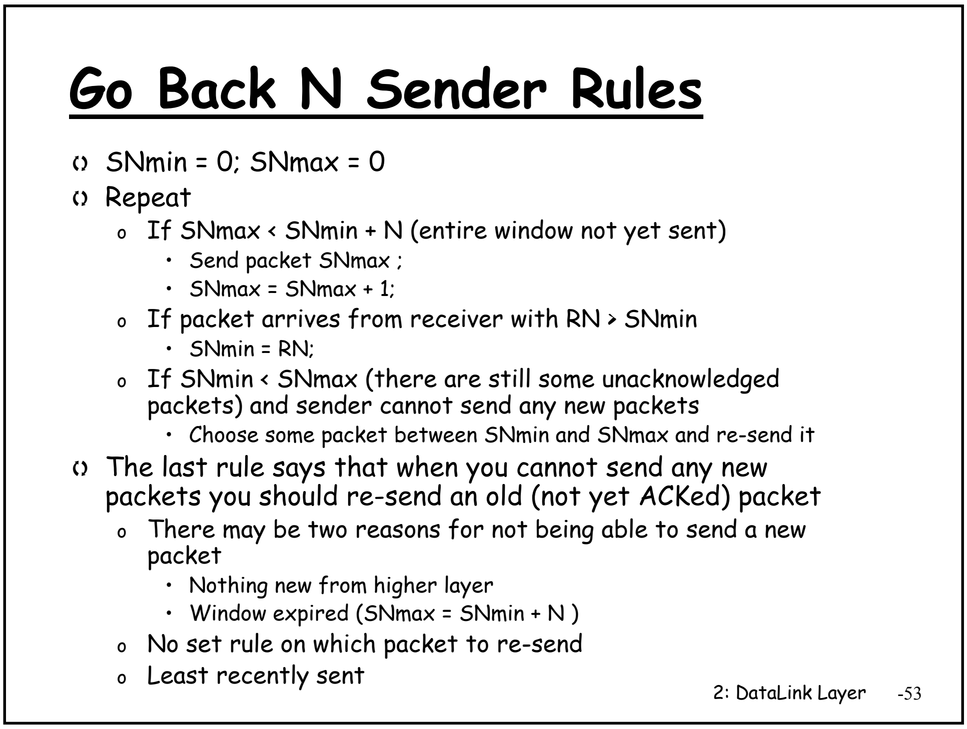
Go-back-N ARQ

- Stop and Wait is inefficient when propagation delay is larger than the packet transmission time
  - Can only send one packet per round-trip time
- Go Back N allows the transmission of new packets before earlier ones are acknowledged
  - Go back N uses a window mechanism where the sender can send packets that are within a "window" (range) of packets
  - The window advances as acknowledgements for earlier packets are received

Features of Go Back N

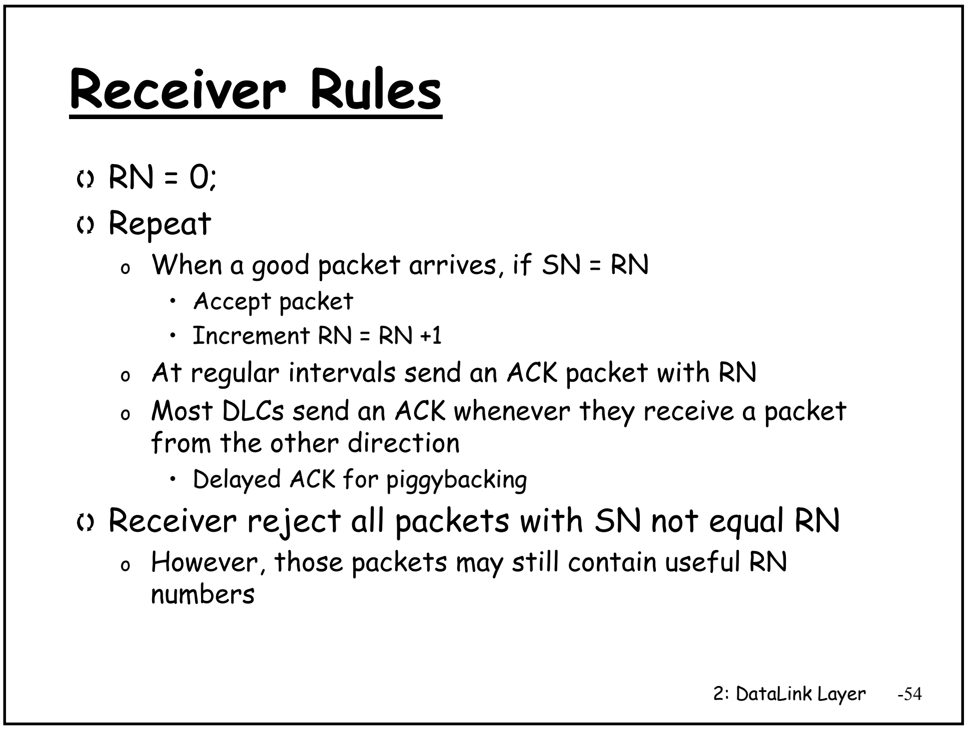
- The receiving entity (B) allocates buffer space for N PDUs
- The transmitting entity (A) is allowed to send N PDUs without waiting for acknowledgment
- Each frame is labeled with a sequence number
- B sends an acknowledgment announcing the next expected PDU
- Sender cannot send packet i+N until it has received the ACK for packet i
- Receiver operates just like in Stop and Wait
  - Receive packets in order
  - Receiver cannot accept packet out of sequence
  - Send RN = i + 1 => ACK for all packets up to and including i
- Use of piggybacking
  - When traffic is bi-directional RN's are piggybacked on packets going in the other direction
  - Each packet contains a SN field indicating that packet's sequence number and a RN field acknowledging packets in the other direction

- The transmitter has a "window" of N packets that can be sent without acknowledgements
  - This window ranges from the last value of RN obtained from the receiver (denoted SNmin) to SNmin+N-1
- When the transmitter reaches the end of its window, or times out, it goes back and retransmits packet SNmin
- Let SNmin be the smallest number packet not yet ACKed
- Let SNmax be the number of the next packet to be accepted from the higher layer (I.e., the next new packet to be transmitted)



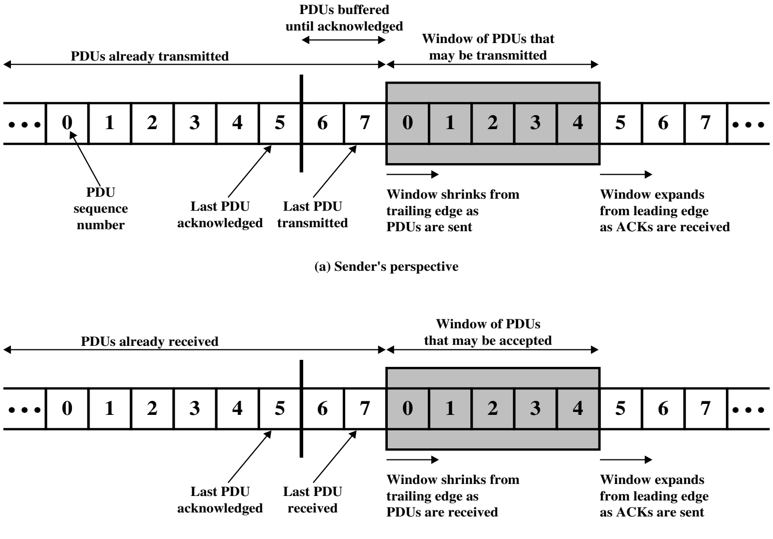
Go Back N Sender Rules

- SNmin = 0; SNmax = 0
- Repeat
  - If SNmax < SNmin + N (entire window not yet sent)
    - Send packet SNmax ;
    - SNmax = SNmax + 1;
  - If packet arrives from receiver with RN > SNmin
    - SNmin = RN;
  - If SNmin < SNmax (there are still some unacknowledged packets) and sender cannot send any new packets
    - Choose some packet between SNmin and SNmax and re-send it
- The last rule says that when you cannot send any new packets you should re-send an old (not yet ACKed) packet
  - There may be two reasons for not being able to send a new packet
    - Nothing new from higher layer
    - Window expired (SNmax = SNmin + N)
  - No set rule on which packet to re-send
  - Least recently sent



Receiver Rules

- RN = 0;
- Repeat
  - When a good packet arrives, if SN = RN
    - Accept packet
    - Increment RN = RN +1
  - At regular intervals send an ACK packet with RN
  - Most DLCs send an ACK whenever they receive a packet from the other direction
    - Delayed ACK for piggybacking
- Receiver reject all packets with SN not equal RN
  - However, those packets may still contain useful RN numbers


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

(a) Sender's perspective

(b) Receiver's perspective

Figure 6.7 Sliding-Window Depiction

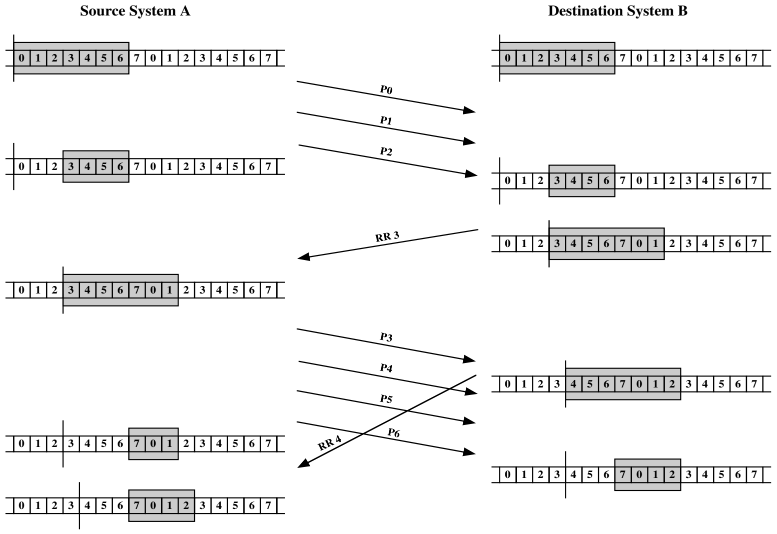


Figure 6.8 Example of a Sliding-Window Protocol

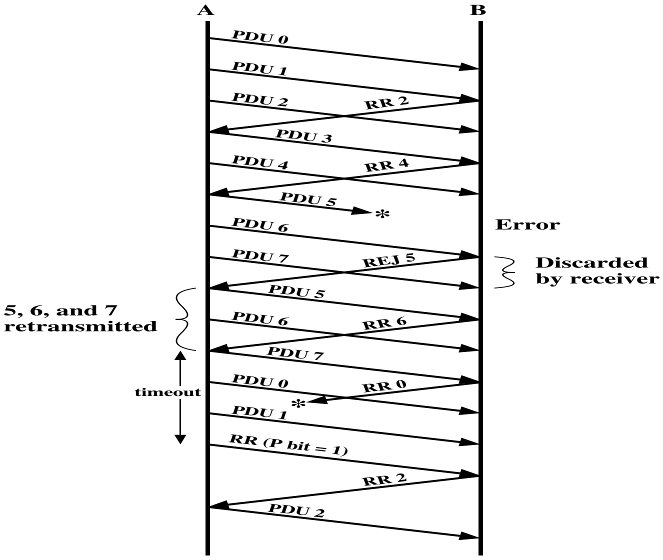


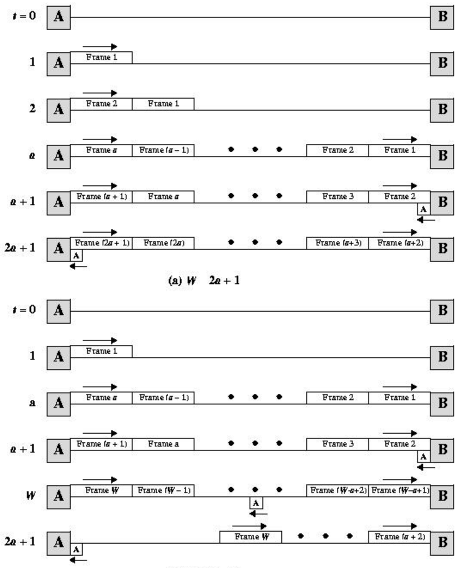
Figure 6.10 Go-back-N ARQ

Go Back N Requirements

- Go Back N is guaranteed to work correctly, independent of the detailed choice of which packets to repeat, if
  - System is correctly initialized
  - No failures in detecting errors
  - Packets travel in FCFS order
  - Positive probability of correct reception
  - Transmitter occasionally resends SNmin (e.g., upon timeout)
  - Receiver occasionally sends RN

Error-Free Sliding Window ARQ

- Case 1: $W \geq 2a + 1$
  - Ack for frame 1 reaches A before A has exhausted its window
- Case 2: $W < 2a + 1$
  - A exhausts its window at $t = W$ and cannot send additional frames until $t = 2a + 1$



(a) $W \quad 2a + 1$

(b) $W < 2a + 1$

Figure 11.10 Timing of Sliding-Window Protocol

Normalized Throughput

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

Go-Back-N ARQ

$N_r = E(\text{number of transmitted frames to successfully transmit one frame})$

$$N_r = \sum_{i=1}^{\infty} f(i) p^{i-1}(1-p)$$

$$= 1 + \sum_{i=1}^{\infty} (i-1) N p^{i-1}(1-p) = 1 - N\sum_{i=1}^{\infty} p^{i-1}(1-p) + N\sum_{i=1}^{\infty} i p^{i-1}(1-p)$$

$$= 1 - N + \frac{N}{1-p} = \frac{1-p+Np}{1-p}$$

$$U = \frac{1 - P}{1 + 2aP} \qquad N = 2a+1, \text{ for } W \geq 2a + 1$$

$$\frac{W(1 - P)}{(2a + 1)(1 - P + WP)} \qquad N=W, \text{ for } W < 2a + 1$$

Notes on Go Back N

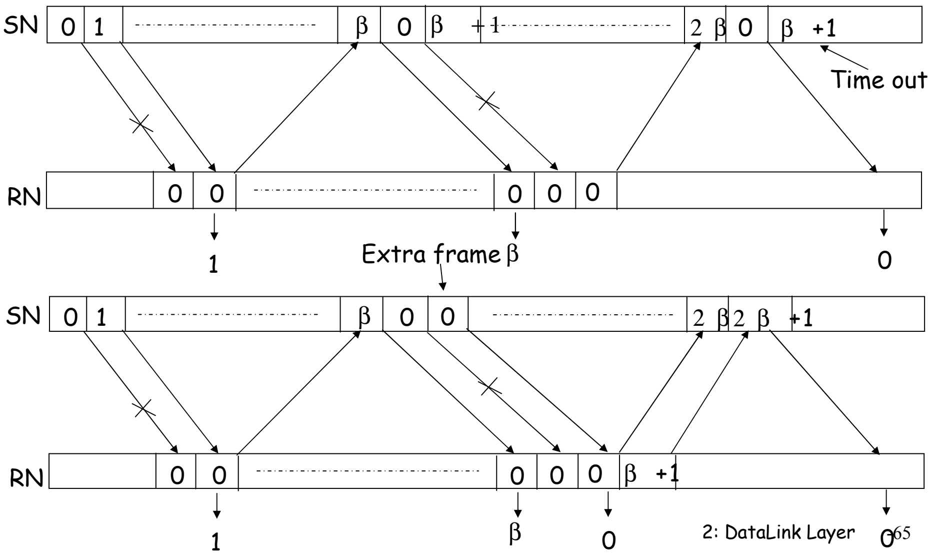
- Requires no buffering of packets at the receiver
- Sender must buffer up to N packets while waiting for their ACK
- Sender must re-send entire window in the event of an error
- Packets can be numbered modulo M where M > N
  - Because at most N packets can be sent simultaneously
- Receiver can only accept packets in order
  - Receiver must deliver packets in order to higher layer
  - Cannot accept packet i+1 before packet i
  - This removes the need for buffering
  - This introduces the need to re-send the entire window upon error
- The major problem with Go Back N is this need to re-send the entire window when an error occurs. This is due to the fact that the receiver can only accept packets in order

Selective Repeat Protocol (SRP)

- Selective Repeat attempts to retransmit only those packets that are actually lost (due to errors)
  - – Receiver must be able to accept packets out of order
  - – Since receiver must release packets to higher layer in order, the receiver must be able to buffer some packets
- Retransmission requests
  - Implicit
    - The receiver acknowledges every good packet, packets that are not ACKed before a time-out are assumed lost or in error
    - Notice that this approach must be used to be sure that every packet is eventually received
  - Explicit
    - An explicit NAK (selective reject) can request retransmission of just one packet
    - This approach can expedite the retransmission but is not strictly needed
  - One or both approaches are used in practice

Selective repeat ARQ with n=2 β+2

Buffer storage is n- β



SRP Rules

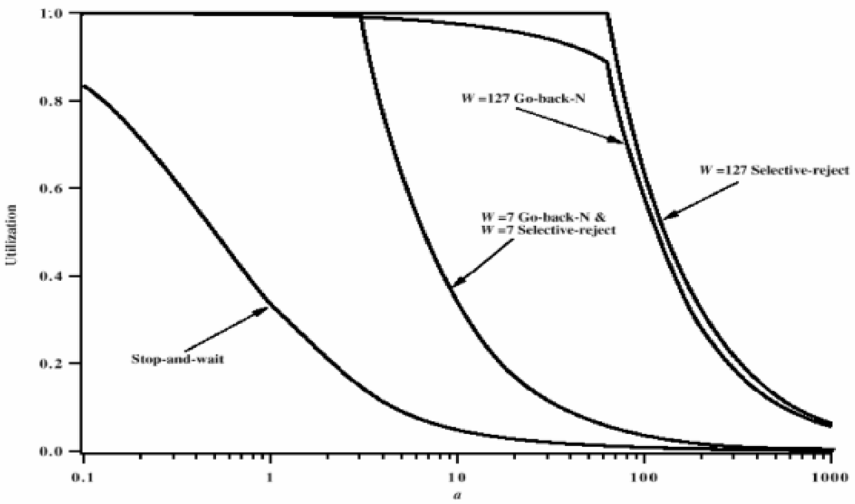
- Window protocol just like GO Back N
  - Window size W
- Packets are numbered Mod M where M >= 2W
- Sender can transmit new packets as long as their number is with W of all un-ACKed packets
- Sender retransmit un-ACKed packets after a timeout
  - Or upon a NAK if NAK is employed
- Receiver ACKs all correct packets
- Receiver stores correct packets until they can be delivered in order to the higher layer

Need for buffering

- Sender must buffer all packets until they are ACKed
  - Up to W un-ACKed packet are possible
- Receiver must buffer packets until they can be delivered in order
  - – I.e., until all lower numbered packets have been received
  - – Needed for orderly delivery of packets to the higher layer
  - – Up to W packets may have to be buffered (in the event that the first packet of a window is lost)
- Implication of buffer size = W
  - Number of un-ACKed packets at sender =< W
- Buffer limit at sender
  - Number of un-ACKed packets at sender cannot differ by more than W
- Buffer limit at the receiver (need to deliver packets in order)
  - Packets must be numbered modulo M >= 2W (using $\log_2(M)$ bits)

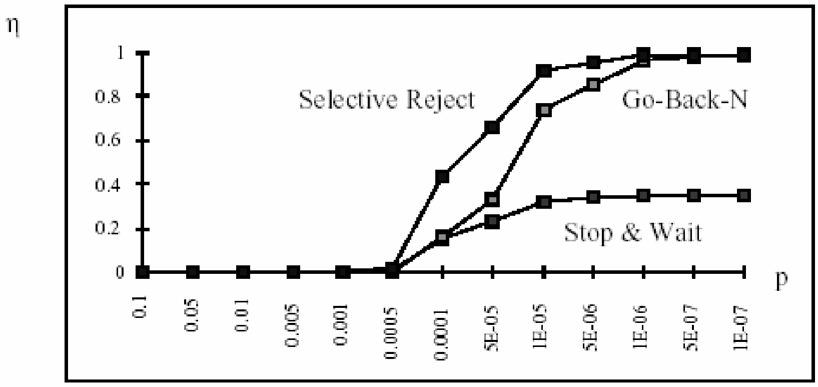
EFFICIENCY

- For ideal SRP, only packets containing errors will be retransmitted
  - – Ideal is not realistic because sometimes packets may have to be retransmitted because their window expired. However, if the window size is set to be much larger than the timeout value then this is unlikely
  - With ideal SRP, efficiency = 1 - P
    - P = probability of a packet error
- When the window size is small performance is about the same, however with a large window SRP is much better
- As transmission rates increase we need larger windows and hence the increased use of SRP



ARQ Utilization as a Function of a (P = 10^{-3})

Transmission Efficiency of ARQ Protocols



Source: Communication Networks by Leon-Garcia 1998

Concurrent Logical Channels (used by ARPANET)

- Multiplex 8 logical channels over a single link
- Run stop-and-wait on each logical channel (but keeps pipe full)
- Maintain three state bits per channel
  - channel busy/not_busy
  - next sequence number in
  - current sequence number out

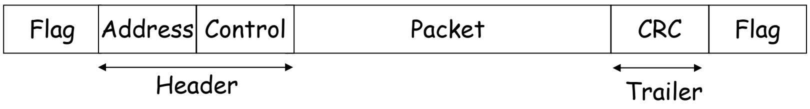
- Header: 3-bit channel num, 1-bit sequence num
  (4-bits total - same as sliding window protocol)

- Separates reliability from order
  (does not keep frames in order and no flow control)

Are there active networking variations of this algorithm that might improve it?

STANDARD DLC's

- HDLC, LAPB (X.25), and SDLC are almost the same
  - HDLC/ SDLC developed by IBM for IBM SNA networks
  - LAPB developed for X.25 networks
- They all use bit oriented framing with flag = 01111110
- They all use a 16-bit CRC for error detection
- They all use Go Back N ARQ with N = 7 or 127 (optional)



Flag	Address	Control	Packet	CRC	Flag

- Older protocols (used for modems, e.g., xmodem) used stop and wait and simple checksums

Point-to-point protocols at the Network layer

- Main functions:
  - Routing
  - Flow control
- Subtle issues:
  - Session Identification (virtual circuits)
  - Packet numbering
  - Window overflow
  - Error recovery
  -

Comparisons

Function	DLC	Network
Error recovery	Host-to-host Smaller delay In-order packet delivery	End-to-end Widely varying delay Not necessarily Floating packets Network Versus Transport Advantages ? disadvantages?
Flow/Congestion control	No need RNR	Adaptive end-to-end window size Delayed ACK to control flow A *permit* (X.25)

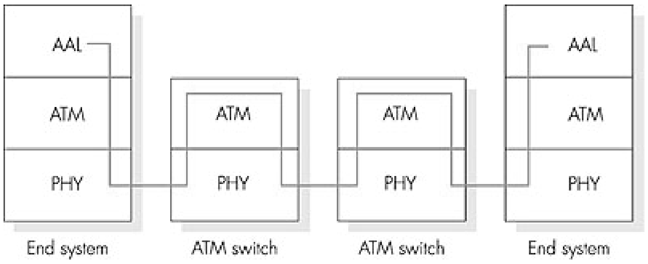
Transport Layer

- Main functions:
  - ERROR CONTROL
  - FLOW/CONGESTION CONTROL
  - SEGMANTATION
  - MULTIPLEXING/DEMULTIPLEXING
- ISO Transport classes
  - Class 0
  - Class 1 (some error recovery)
  - Class 2 (multiplexing, but no error recovery)
  - Class 3 (multiplexing and no error recovery)
  - Class 4 (similar to TCP)

Asynchronous Transfer Mode: ATM

- **1990's/00 standard for high-speed** (155Mbps to 622 Mbps and higher) *Broadband Integrated Service Digital Network* architecture
- Goal: *integrated, end-end transport of carry voice, video, data*
  - meeting timing/QoS requirements of voice, video (versus Internet best-effort model)
  - "next generation" telephony: technical roots in telephone world
  - packet-switching (fixed length packets, called "cells") using virtual circuits

ATM architecture



- **adaptation layer: only at edge of ATM network**
  - data segmentation/reassembly
  - roughly analogous to Internet transport layer
- ATM layer: "network" layer
  - cell switching, routing
- physical layer

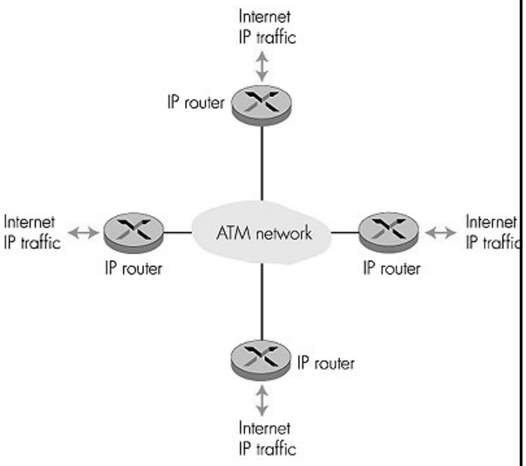
ATM: network or link layer?

Vision: end-to-end transport: "ATM from desktop to desktop"

- ATM *is* a network technology

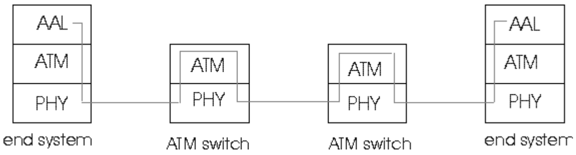
Reality: used to connect IP backbone routers

- "IP over ATM"
- ATM as switched link layer, connecting IP routers



ATM Adaptation Layer (AAL)

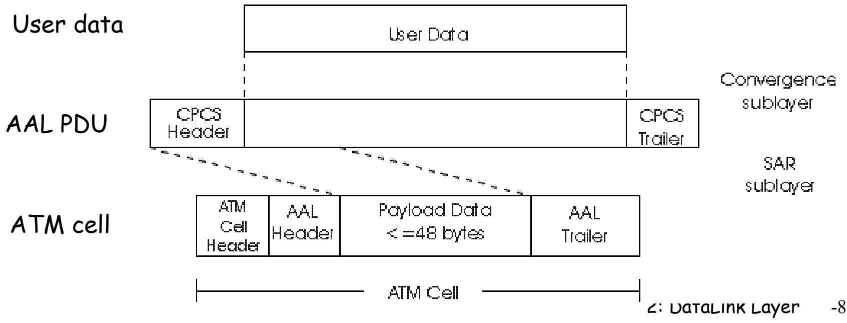
- ATM **Adaptation Layer** (AAL): “adapts” upper layers (IP or native ATM applications) to ATM layer below
- AAL present **only in end systems**, not in switches
- AAL layer segment (header/trailer fields, data) fragmented across multiple ATM cells
  - analogy: TCP segment in many IP packets



ATM Adaptation Layer (AAL) [more]

Different versions of AAL layers, depending on ATM service class:

- AAL1: for CBR (Constant Bit Rate) services, e.g. circuit emulation
- AAL2: for VBR (Variable Bit Rate) services, e.g., MPEG video
- AAL5: for data (eg, IP datagrams)



AAL5 - Simple And Efficient AL (SEAL)

- **AAL5: low overhead** AAL used to carry IP datagrams
  - 4 byte cyclic redundancy check
  - PAD ensures payload multiple of 48bytes
  - large AAL5 data unit to be fragmented into 48-byte ATM cells

CPCS-PDU payload	PAD	Length	CRC
0-65535	0-47	2	4

ATM Layer

Service: transport cells across ATM network

- analogous to IP network layer
- very different services than IP network layer

Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

ATM Layer: Virtual Circuits

- VC transport: cells carried on VC from source to dest
  - call setup, teardown for each call *before* data can flow
  - each packet carries VC identifier (not destination ID)
  - *every* switch on source-dest path maintain "state" for each passing connection
  - link,switch resources (bandwidth, buffers) may be *allocated* to VC: to get circuit-like perf.
- Permanent VCs (PVCs)
  - long lasting connections
  - typically: "permanent" route between to IP routers
- Switched VCs (SVC):
  - dynamically set up on per-call basis

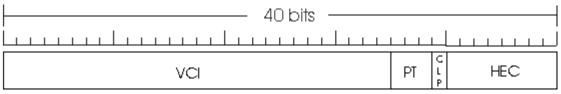
ATM VCs

- Advantages of ATM VC approach:
  - QoS performance guarantee for connection mapped to VC (bandwidth, delay, delay jitter)
- Drawbacks of ATM VC approach:
  - Inefficient support of datagram traffic
  - one PVC between each source/dest pair) does not scale (N*2 connections needed)
  - SVC introduces call setup latency, processing overhead for short lived connections

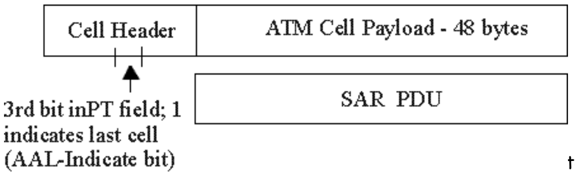
ATM Layer: ATM cell

- 5-byte ATM cell header
- 48-byte payload
  - Why?: small payload -> short cell-creation delay for digitized voice
  - halfway between 32 and 64 (compromise!)

Cell header

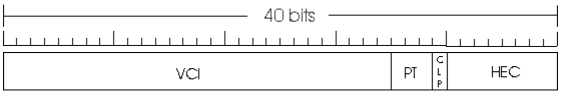


Cell format



ATM cell header

- **VCI:** virtual channel ID
  - will *change* from link to link thru net
- **PT:** Payload type (e.g. RM cell versus data cell)
- **CLP:** Cell Loss Priority bit
  - CLP = 1 implies low priority cell, can be discarded if congestion
- **HEC:** Header Error Checksum
  - cyclic redundancy check



ATM Physical Layer (more)

Two pieces (sublayers) of physical layer:

- Transmission Convergence Sublayer (TCS): adapts ATM layer above to PMD sublayer below
- Physical Medium Dependent: depends on physical medium being used

TCS Functions:

  - Header **checksum** generation: 8 bits CRC
  - Cell **delineation**
  - With "unstructured" PMD sublayer, transmission of **idle cells** when no data cells to send

ATM Physical Layer

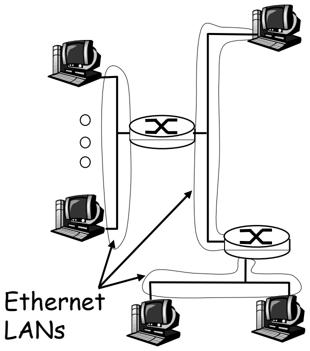
Physical Medium Dependent (PMD) sublayer

- **SONET/SDH**: transmission frame structure (like a container carrying bits);
  - bit synchronization;
  - bandwidth partitions (TDM);
  - several speeds: OC3 = 155.52 Mbps; OC12 = 622.08 Mbps; OC48 = 2.45 Gbps, OC192 = 9.6 Gbps
- **TI/T3**: transmission frame structure (old telephone hierarchy): 1.5 Mbps/ 45 Mbps
- **unstructured**: just cells (busy/idle)

IP-Over-ATM

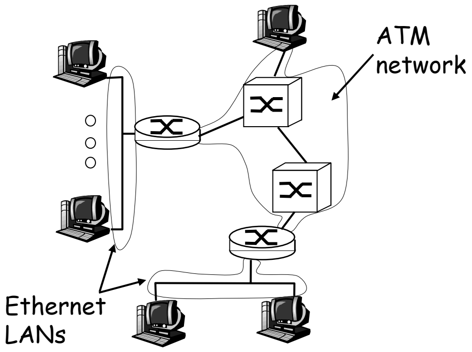
Classic IP only

- 3 "networks" (e.g., LAN segments)
- MAC (802.3) and IP addresses



IP over ATM

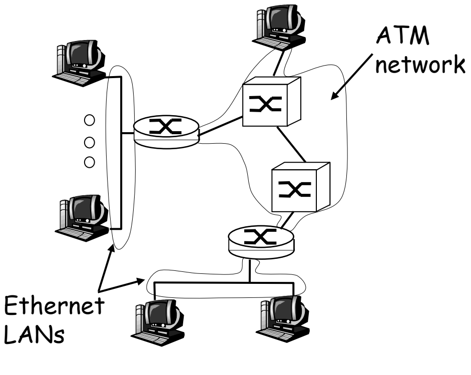
- replace "network" (e.g., LAN segment) with ATM network
- ATM addresses, IP addresses



IP-Over-ATM

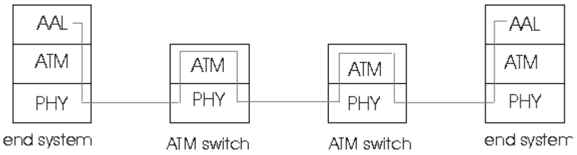
Issues:

- IP datagrams into ATM AAL5 PDUs
- from IP addresses to ATM addresses
  - just like IP addresses to 802.3 MAC addresses!



Datagram Journey in IP-over-ATM Network

- at Source Host:
  - IP layer maps between IP, ATM dest address (using ARP)
  - passes datagram to AAL5
  - AAL5 encapsulates data, segments cells, passes to ATM layer
- ATM network: moves cell along VC to destination
- at Destination Host:
  - AAL5 reassembles cells into original datagram
  - if CRC OK, datagram is passed to IP



ARP in ATM Nets

- ATM network needs destination ATM address
  - just like Ethernet needs destination Ethernet address
- IP/ATM address translation done by ATM ARP (Address Resolution Protocol)
  - ARP server in ATM network performs broadcast of ATM ARP translation request to all connected ATM devices
  - hosts can register their ATM addresses with server to avoid lookup

X.25 and Frame Relay

Like ATM:

- wide area network technologies
- Virtual-circuit oriented
- origins in telephony world
- can be used to carry IP datagrams
  - can thus be viewed as Link Layers by IP protocol

X.25

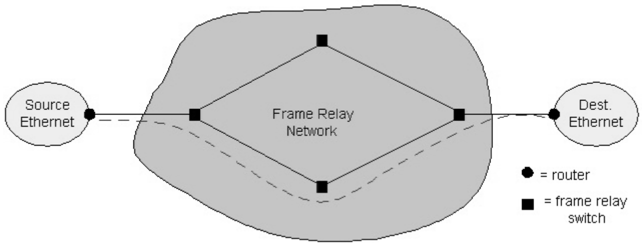
- X.25 builds VC between source and destination for each user connection
- Per-hop control along path
  - error control (with retransmissions) on each hop using LAP-B
    - variant of the HDLC protocol
  - per-hop flow control using credits
    - congestion arising at intermediate node propagates to previous node on path
    - back to source via back pressure

IP versus X.25

- X.25: reliable in-sequence end-end delivery from end-to-end
  - "intelligence in the network"
- IP: unreliable, out-of-sequence end-end delivery
  - "intelligence in the endpoints"
- gigabit routers: limited processing possible
- 2000: IP wins

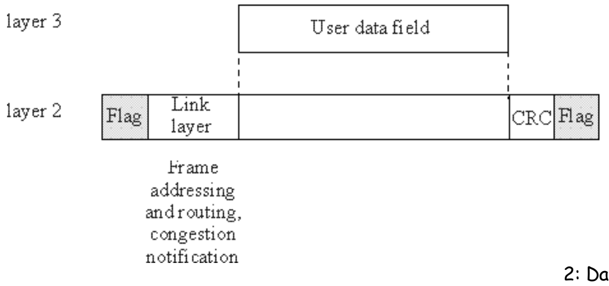
Frame Relay

- Designed in late '80s, widely deployed in the '90s
- Frame relay service:
  - no error control
  - end-to-end congestion control



Frame Relay (more)

- Designed to **interconnect** corporate customer LANs
  - typically permanent VC's: "**pipe**" carrying aggregate traffic between two routers
  - switched VC's: as in ATM
- corporate customer **leases** FR service from public Frame Relay network (eg, Sprint, ATT)



Frame Relay (more)

flags	address	data	CRC	flags

- Flag bits, 01111110, delimit frame
- address:
  - 10 bit VC ID field
  - 3 congestion control bits
    - FECN: forward explicit congestion notification (frame experienced congestion on path)
    - BECN: congestion on reverse path
    - DE: discard eligibility

Frame Relay -VC Rate Control

- Committed Information Rate (CIR)
  - defined, “guaranteed” for each VC
  - negotiated at VC set up time
  - customer pays based on CIR
- DE bit: Discard Eligibility bit
  - Edge FR switch measures traffic rate for each VC; marks DE bit
  - DE = 0: high priority, rate compliant frame; deliver at “all costs”
  - DE = 1: low priority, eligible for congestion discard

Frame Relay - CIR & Frame Marking

- **Access Rate**: rate **R** of the access link between **source router** (customer) and **edge FR switch** (provider); 64Kbps < **R** < 1,544Kbps
- Typically, **many VCs** (one per destination router) multiplexed on the same access trunk; each VC has own **CIR**
- Edge FR switch **measures** traffic rate for each VC; it **marks**
- (ie DE <= 1) frames which **exceed** CIR (these may be later dropped)