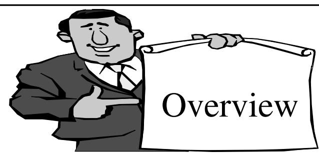
Data Link Layer

Shivkumar Kalyanaraman
Rensselaer Polytechnic Institute
shivkuma@ecse.rpi.edu
http://www.ecse.rpi.edu/Homepages/shivkuma

Based in part upon the slides of Prof. Raj Jain (OSU)



- The data link layer problem
- Error detection and correction
- Simple datalink protocols
- Sliding window protocols
- Example datalink protocols

Links

Sometimes you install your own

Category 5 twisted pair	10-100Mbps, 1
50-ohm coax (ThinNet)	10-100Mbps, 2
75-ohm coax (ThickNet)	10-100Mbps, 5
Multimode fiber	100Mbps, 2km
Single-mode fiber	100-2400Mbps

Sometimes leased from the phone company

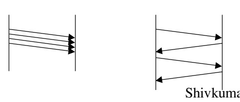
Service to ask for	Bandwidth you get
ISDN	64 Kbps
T1	1.544 Mbps
T3	44.736 Mbps
STS-1	51.840 Mbps
STS-3	155.250 Mbps
STS-12	622.080 Mbps
STS-24	1.244160 Gbps
STS-48	2.488320 Gbps

The data link layer problem

- How to provide a ***"reasonable" link abstraction*** to the higher layers ?
  - I.e., accept "frames" from many higher layer protocols and compensate for link errors or lack of destination buffers …
  - Other types of "abstractions" possible too
- Problems:
  - Framing, protocol multiplexing
  - Error Control
  - Flow Control

Datalink Layer Services

- Unacknowledged connectionless service
  - No acks, no connection
  - Error recovery up to higher layers
  - For low error-rate links or voice traffic
- Acknowledged connectionless service
  - Acks improve reliability
  - For unreliable channels. E.g.: Wireless Systems



Datalink Services (Cont)

- Acknowledged connection-oriented service
  - Equivalent of reliable bit-stream
  - Connection establishment
  - Packets Delivered In-Order
  - Connection Release
  - Inter-Router Traffic
- Typically implemented by network adaptor
  - Adaptor fetches (deposits) frames out of (into) host memory

Framing

- Framing = How to break a bit-stream into frames
- Need for framing: Error Detection/Control work on chunks and not on bit streams of data
- Framing methods:
  - ***Timing*** : risky. No network guarantees.
  - ***Character count:*** may be garbled by errors
  - ***Character delimiter and stuffing:*** Delimit frame with special characters
  - ***Bit stuffing:*** delimit frame with bit pattern
  - Physical layer ***coding violations***

Character Stuffing

- Delimit with DLE STX or DLE ETX character flags
- Insert 'DLE' before accidental 'DLE' in data
- Remove stuffed character at destination

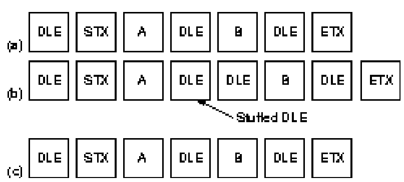


Fig 3-4

Bit Stuffing

- Delimit with special bit pattern (bit flags)
- Stuff bits if pattern appears in data
- Remove stuffed bits at destination

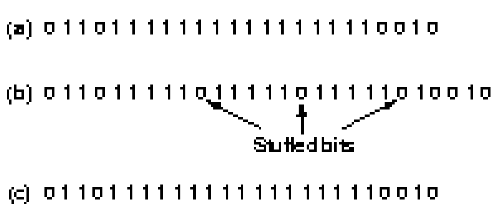
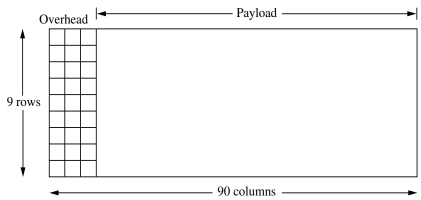


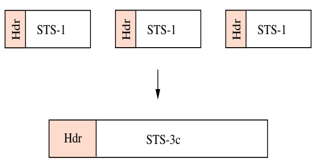
Fig 3-5

Clock-Based Framing

- SONET: Synchronous Optical Network
- ITU standard for transmission over fiber
- STS-1 (51.84 Mbps)



- Byte-interleaved multiplexing



- Each frame is 125µs long (remember, 1/8000 ?)

Error Detection and Correction

- Single bit-errors vs Burst Errors
  110101 → 100101 vs 100001
- n-bit codeword = m message bits + r check bits
- Hamming Distance = # of different bits
  1010101
  <u>1001010</u>
  0011111 ⇒ Hamming distance = 5
- Distance d code = minimum Hamming distance between any two code words written in the code
- To detect d-bit errors, distance d+1 code required
- To correct d-bit errors, distance 2d+1 code required

Parity Checks

1	0	1	1	1	0	1	0	
1	2	3	4	5	6	7	8	9

- Odd Parity

1	0	1	1	1	0	1	0	0
1	2	3	4	5	6	7	8	9

0	0	1	1	1	0	1	0	0
1	2	3	4	5	6	7	8	9

1-bit error

0	0	0	1	0	0	1	0	0
1	2	3	4	5	6	7	8	9

3-bit error

0	0	0	1	1	0	1	0	0
1	2	3	4	5	6	7	8	9

2-bit error

- Even Parity

1	0	1	1	1	0	1	1	0
1	2	3	4	5	6	7	8	9

Check Digit Method

- Make number divisible by 9

Example: 823 is to be sent

1. Left-shift: 8230
2. Divide by 9, find remainder: 4
3. Subtract remainder from 9: 9-4=5
4. Add the result of step 5 to step 1: 8235
5. Check that the result is divisible by 9.

Detects all single-digit errors: <u>7</u>235, 8<u>3</u>35, 82<u>5</u>5, 823<u>7</u>

Detects several multiple-digit errors: 8<u>76</u>5, <u>7346</u>

Does not detect some errors: <u>73</u>35, 8<u>77</u>5, ...

Modulo 2 Arithmetic

```
  1111      11001
 +1010   ×     11
 --------  --------
  0101      11001
           11001
           ---------
           101011
```

```
       1100
 11  | 10101
     / 11
     -----
       x11
        11
      -----
        x00
         00
       -----
         x01
          00
         ----
          x1
```

```
010   2
011   3
----  --
001   1  Mod 2
101   5  Binary
```

Cyclic Redundancy Check (CRC)

- ***Binary Check Digit Method***
- Make number divisible by P=110101 (n+1=6 bits)

Example: M=1010001101 is to be sent

1. Left-shift M by n bits 2^nM= 101000110100000
2. Divide 2^nM by P, find remainder: R=01110
3. ~~Subtract remainder from P~~ ← Not required in Mod 2
4. Add the result of step 2 to step 1 : T=101000110101110
5. Check that the result T is divisible by P.

Modulo 2 Division

```
      Q=1101010110
P=110101)101000110100000=2^nM
         110101
          111011                010110
          110101                000000
           011101                101100
           000000                110101
            111010                110010
            110101                110101
             011111                001110
             000000                000000
              111110                01110 = R
              110101
```


Checking At The Receiver

```
       1101010110
110101)101000110101110
       110101
        111011                010111
        110101                000000
         011101                101111
         000000                110101
          111010                110101
          110101                110101
           011111                00000
           000000
            111110
            110101
```

Polynomial Representation

- Number the bits 0, 1, ..., from <u>right</u>
  $b_n b_{n-1} b_{n-2} \ldots b_3 b_2 b_1 b_0$
  $b_n x^n + b_{n-1} x^{n-1} + b_{n-2} x^{n-2} + \ldots + b_3 x^3 + b_2 x^2 + b_1 x + b_0$
- Example:
  543210
  ↓↓↓↓↓↓
  $110101 = x^5+x^4+x^2+1$
  $1101\ 1001\ 0011 = x^{11}+x^{10}+x^8+x^7+x^4+x+1$
  11 10 9 8 1 0

Cyclic Redundancy Check (CRC)

Polynomial Division Method

Make $T(x)$ divisible by $P(x) = x^5+x^4+x^2+1$ (Note: n=5)

Example: M=1010001101 is to be sent
$M(x) = x^9+x^7+x^3+x^2+1$

1. Multiply M(x) by x^n, $x^nM(x) = x^{14}+x^{12}+x^8+x^7+x^5+$ ....
2. Divide $x^nM(x)$ by $P(x)$, find remainder: $R(x)$=01110=x^3+x^2+x

CRC (Cont)

- 3. Add the remainder $R(x)$ to $x^nM(x)$:
  $T(x) = x^{14}+x^{12}+x^8+x^7+x^5+x^3+x^2+x$
- 4. Check that the result $T(x)$ is divisible by $P(x)$.
- Transmit the bit pattern corresponding to $T(x)$:
  101000110101110

Popular CRC Polynomials

- CRC-12: $x^{12}+x^{11}+x^3+x^2+x+1$
- CRC-16: $x^{16}+x^{15}+x^2+1$
- CRC-CCITT: $x^{16}+x^{12}+x^5+1$
- CRC-32: Ethernet, FDDI, ...
  $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}$
  $+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

Even number of terms in the polynomial
$\Rightarrow$ Polynomial is divisible by 1+x
$\Rightarrow$ Will detect all odd number of bit errors

Error Correcting Codes (ECC)

- Example: VRC+LRC will correct all single bit errors
- Forward error correction (FEC)
  Used if retransmission expensive

Vertical Redundancy Check

```
1 0 1 1 0 1 1 1
1 1 0 1 0 1 1 1
0 0 1 1 1 0 1 0
1 1 1 1 0 0 0 0
1 0 0 0 1 0 1 1
0 1 0 1 1 1 1 1
0 1 1 1 1 1 1 0
```

Longitudinal Redundancy Check

Fig 4.5b Stallings

Error-Correcting Codes

- Enough redundant information in a frame to detect and correct the error
- Lower limit on number of check bits to correct 1 error: $(m+r+1) \le 2^r$
- Hamming's method: (corrects 1-bit errors)
  - Check bits in positions in powers of 2 (1,2,4 ...)
  - Each data bit included in several check bits
    E.g., data bit 11 included in check bits 1, 2, 8
  - Other bit positions data

Burst Error Correction

- Arrange code words as matrix
- Transmit one column at a time
- Uses kr check bits to immunize blocks of km data bits to single burst error of length k or less

Char.	ASCII	Check bits
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	11111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	00101011111
d	1100100	11111001100
e	1100101	00111000101

Order of bit transmission

Fig 3-6

Internet Checksum Algorithm

Idea: view message as a sequence of 16-bit integers. Add these integers together using 16-bit ones complement arithmetic, and then take the ones complement of the result. That 16-bit number is the checksum.

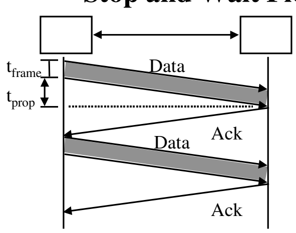
Flow Control

- Flow Control = Sender does not flood the receiver, but maximizes throughput
- Sender throttled until receiver grants permission

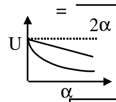
Sliding Window Protocols

- Window = Set of sequence numbers to send/receive
- Sender window
  - Sender window increases when ack received
  - Packets in sender window must be buffered at source
  - Sender window may grow in some protocols

Stop and Wait Flow Control



$$U = \frac{t_{frame}}{2t_{prop}+t_{frame}} = \frac{1}{2\alpha + 1}$$



Light in vacuum = 300 m/µs
Light in fiber = 200 m/µs
Electricity = 250 m/µs

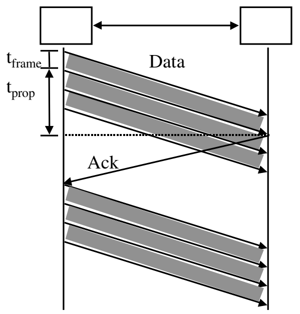
$$\alpha = \frac{t_{prop}}{t_{frame}} = \frac{\text{Distance/Speed of Signal}}{\text{Frame size /Bit rate}} = \frac{\text{Distance} \times \text{Bit rate}}{\text{Frame size} \times \text{Speed of Signal}}$$

Utilization: Examples

- ***Satellite Link:*** Propagation Delay t_{prop} = 270 ms
  Frame Size = 4000 bits = 500 bytes
  Data rate = 56 kbps $\Rightarrow t_{frame}$ = 4/56 = 71 ms
  $\alpha = t_{prop}/t_{frame}$ = 270/71 = 3.8
  U = 1/(2α+1) = ***0.12*** (too low !!)

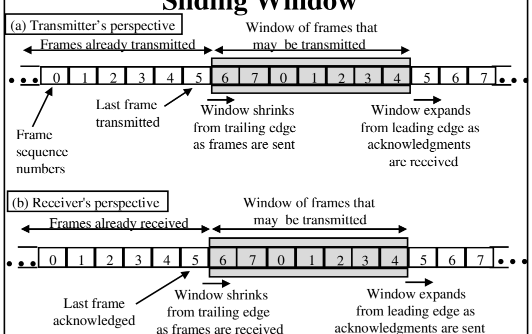
- ***Short Link:*** 1 km = 5 µs,
  Rate=10 Mbps,
  Frame=500 bytes $\Rightarrow t_{frame}$= 4k/10M= 400 µs
  $\alpha=t_{prop}/t_{frame}$=5/400=0.012 ⇒ U=1/(2α+1)= ***0.98*** (great!)

Sliding Window Protocol

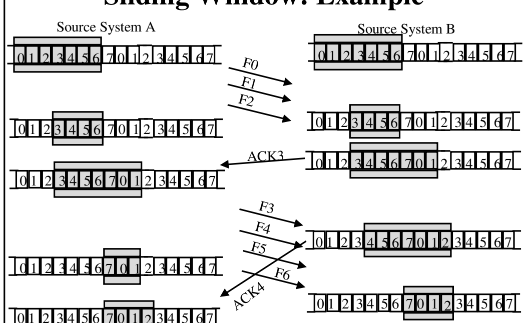


$$U = \frac{N t_{frame}}{2t_{prop} + t_{frame}} = \begin{cases} \dfrac{N}{2\alpha+1} \\ 1 \text{ if } N > 2\alpha+1 \end{cases}$$

Sliding Window



Sliding Window: Example



Effect of Window Size

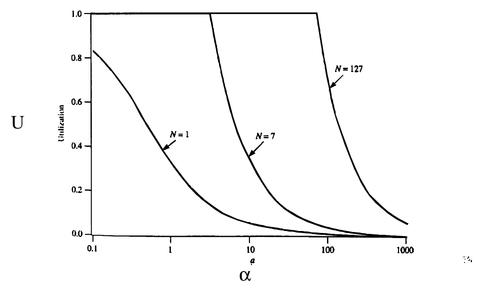
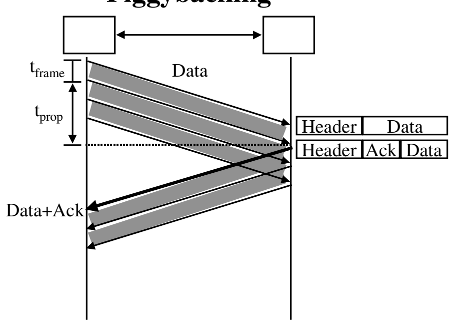


Fig 5.9 Stallings

Piggybacking

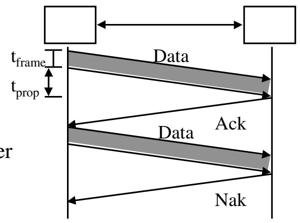


Packet-error Control

- Error Control = Deliver frames without error, in the proper order to network layer
- Error control Mechanisms:
  - ***Ack/Nak:*** Provide sender some feedback about other end
  - ***Time-out:*** for the case when entire packet or ack is lost
  - ***Sequence numbers:*** to distinguish retransmissions from originals, and to identify what is acked/nacked

Packet-error Control

- Automatic Repeat Request (ARQ)
  - Error detection
  - Acknowledgment
  - Retransmission after timeout
  - Negative Acknowledgment



Stop-and-Wait ARQ

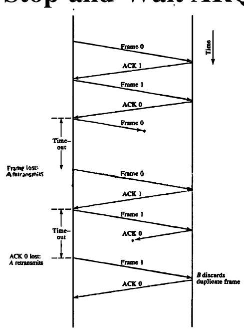
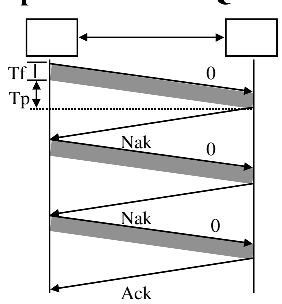


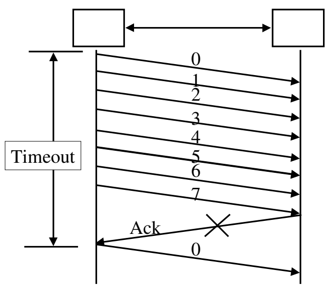
Fig 5.10

Performance: Stop-and-Wait ARQ

- P=Probability of Frame Error
- $\alpha = T_p/T_f$
- $U=T_f/[N_r(T_f+2T_p)]$ $= 1/[N_r(1+2\alpha)]$
- $N_r=\Sigma\ i\ P^{i-1}(1-P)$ $=1/(1-P)$
- **$U=(1-P)/(1+2\alpha)$**

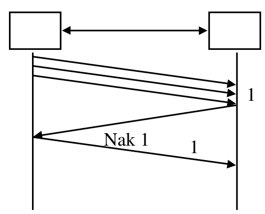


Selective Repeat: Window Size



Sequence number space $\geq$ 2 window size

Performance: Selective Repeat



- Error Free:
  U=1 if $N>2\alpha+1$
  $N/(2\alpha+1)$ otherwise
- With Errors:
  $N_r = \Sigma\, i\, P^{i-1}(1-P)$
  $=1/(1-P)$
- **$U=(1-P)$ if $N>(1+2\alpha)$**
  $N(1-P)/(1+2\alpha)$ otherwise

Note that we have simply divided the error-free sliding window result by Nr, where, Nr = E [# transmitted frames to successfully transmit one frame] = 1-P in this case...

Go Back n: Example

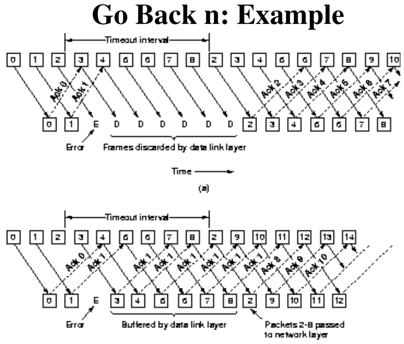
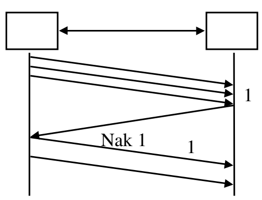


Fig 3-15

Go-back-N

- Damaged Frame
  - Frame received with error
  - Frame lost
  - Last frame lost
- Damaged Ack
  - One ack lost, next one makes it
  - All acks lost
- Damaged Nak

Performance: Go-back-N



- Frames Retransmitted = $2\alpha+1$ if $N>2\alpha+1$
  N otherwise
- $U = (1-P)/(1+2\alpha P)$ if $N>2\alpha+1$
  $N(1-P)/[(2\alpha+1)(1-P+NP)]$ otherwise

Selective-Repeat ARQ

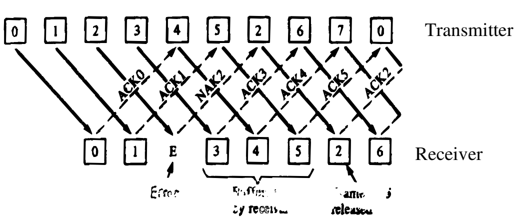


Fig 5.11b

Performance Comparison

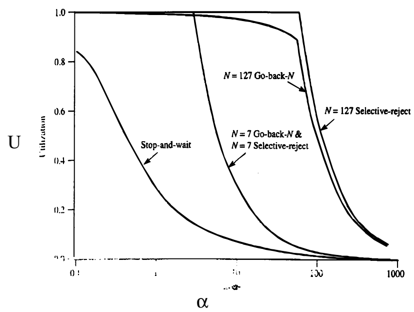


Fig 5.12

HDLC Family

- Synchronous Data Link Control (SDLC): IBM
- **High-Level Data Link Control (HDLC): ISO**
- Link Access Procedure-Balanced (LAPB): X.25
- Link Access Procedure for the D channel (LAPD): ISDN
- Link Access Procedure for modems (LAPM): V.42
- Link Access Procedure for half-duplex links (LAPX): Teletex
- **Point-to-Point Protocol (PPP): Internet**
- **Logical Link Control (LLC): IEEE**
- Advanced Data Communications Control Procedures (ADCCP): ANSI
- V.120 and Frame relay also use HDLC

HDLC

- High-level Data Link Control
- Bit-oriented, bit-stuffing for transparency
- Control field for seq numbers, acks etc
- Data field arbitrarily long (practical limits: checksum efficiency)
- CRC using CRC-CCITT generator polynomial

Bits	8	8	8	≥0	16	8
	01111110	Address	Control	Data	Checksum	01111110

HDLC Frame Types

- Information, Supervisory, Unnumbered frames
- Sliding window with 3-bit sequence number, piggybacked acks
- New: Conn'n Management through supervisory frames

Bits	1	3		1	3
(a)	0	Seq		P/F	Next
(b)	1	0	Type	P/F	Next
(c)	1	1	Type	P/F	Modifier

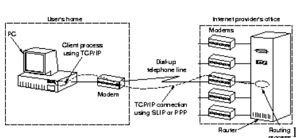
Fig 3-25

HDLC Frames

- Information Frames: User data
  - Piggybacked Acks: Next frame expected
  - Poll/Final = Command/Response
- Supervisory Frames: Flow and error control
  - Go back N and Selective Reject
  - Final ⇒ No more data to send
- Unnumbered Frames: Control (*bla, bla, bla …*)
  - Mode setting commands and responses
  - Information transfer commands and responses
  - Recovery commands and responses
  - Miscellaneous commands and responses

Data Link Layer in the Internet

- Point-to-Point link scenarios
- Router-router leased line (PPP)
- dial-up host-router connection (old SLIP, now PPP)
- SLIP: Not an approved Internet Standard ⇒ incompatibilities
- PPP: Official Internet Standard.



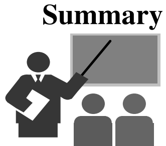
Serial IP (SLIP)

- Simple: only framing = Flags + byte-stuffing
- Compressed headers (CSLIP) for efficiency on low speed links for interactive traffic.
- Problems:
  - Need other end's IP address a priori (can't dynamically assign IP addresses)
  - No "type" field => no multi-protocol encapsulation
  - No checksum => all errors detected/corrected by higher layer.
- RFCs: 1055, 1144

PPP

- Frame format similar to HDLC
- Multi-protocol encapsulation, CRC, dynamic address allocation possible
  - key fields: flags, protocol, CRC (fig 2.3)
- Asynchronous and synchronous communications possible
- Link and Network Control Protocols (LCP, NCP) for flexible control & peer-peer negotiation
- Can be mapped onto low speed (9.6Kbps) and high speed channels (SONET)
- RFCs: 1548, 1332

Summary



- The data link layer problem
- Framing methods
- Error and Flow Control: Sliding Window
- HDLC, SLIP, PPP.