Data Link Control

#8

PowerPoint Section	Page numbers and chapters from McGraw-Hill Create book	Page numbers and chapters from Data Communications and Networking 5th Edition
8-Data Link	Chapter 14 Chapter 16 pp 491 to pp 511 Chapter 15 Chapter 8 Traffic Shaping or Policing pp 234 to pp 238	Chapter 9 Section 23.2 pp 707 to pp 727 Chapter 11 Section 30.2.2 Traffic Shaping or Policing pp 1058 to pp 1062

Outline

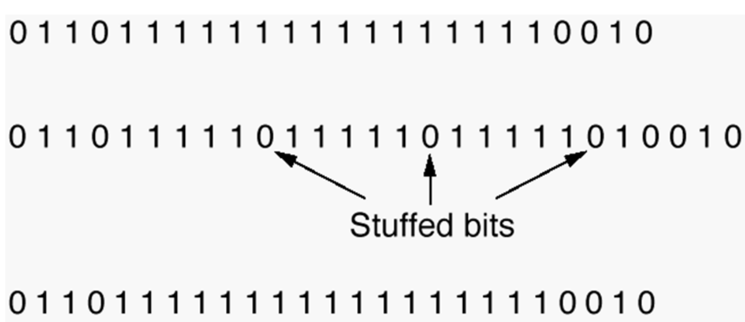
- DLC functions
- DLC Framing
- Error and flow control
- Performance of DLC
- Example of a standard DLC protocol->HDLC
- Open loop flow control

Data Link Layer Functions

- Data Link layer provides a 'error free' point-to-point bit pipe for transmission of network layer PDU's.
  - Framing
  - Error Control
  - Flow Control
  - Error Detection

Framing

- Flags
  - Insert special bit patterns, called 'flags' at start and end of the frame.
    - 01111110

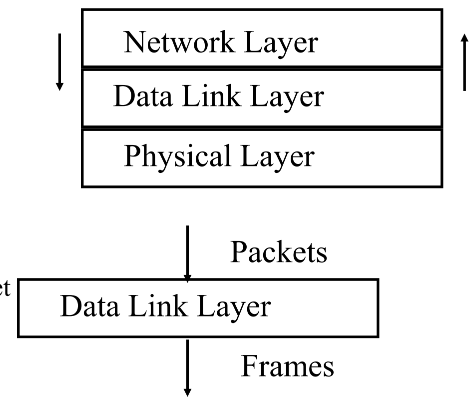


From: "Computer Networks, 3rd Edition, A.S. Tanenbaum. Prentice Hall, 1996

Error and Flow Control

Network and data link layers only communicate via messages with specific data structures.

The data link layer processes those structures with a set of procedures.



Error and Flow Control

Required procedures

- FromNetworkLayer
  - Fetch information from the network layer
- ToNetworkLayer
  - Deliver information to the network layer
- FromPhysicalLayer
  - Fetch information from the physical layer
- ToPhysicalLayer
  - Deliver information to the physical layer

Error and Flow Control

Required procedures

- Timers
  - StartTimer
  - StopTimer
  - StartAckTimer
  - StopAckTimer
- EnableNetworkLayer
  - Turn on flow of information from the network layer
- DisableNetworkLayer
  - Turn off flow of information from the network layer

Error and Flow Control

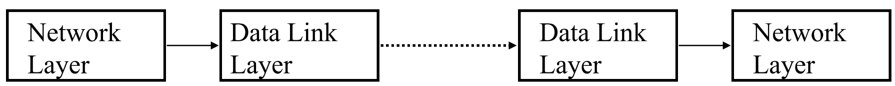
Events

- Networks are Asynchronous
  - Arrival time of packet and acknowledgments are unknown
- Arrival of packet and acknowledgments triggers some action by the protocol
  - Action is a function of the type of arrival
  - State of the protocol
- Examples:
  - FrameArrival
  - CksumErr (detected error)

Error and Flow Control

Protocol 1: The Unrestricted Simplex Protocol

- Assumptions
  - One directional information flow
  - Infinite buffers
  - No errors
  - Network Layer always has a packet to send



Error and Flow Control

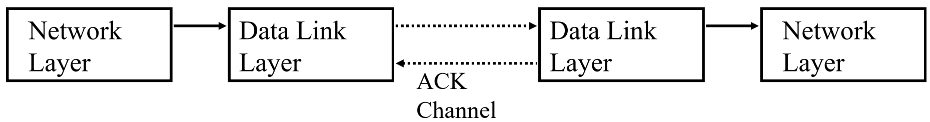
Protocol 2:
The Simplex Stop & Wait Protocol: Assumptions

- One directional information flow
- No errors
- Network Layer always has a packet to send
- Finite receive buffers
  - Finite buffer means that there must be some way to stop the transmitter from sending when the buffer is full

Error and Flow Control

Protocol 2: The Simplex Stop & Wait Protocol

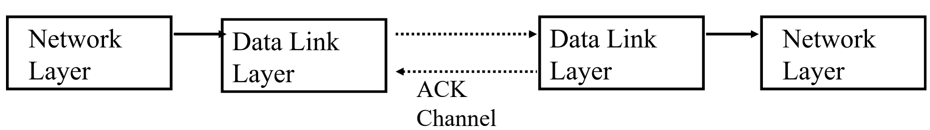
Assume Network
Layer always has
data to send



Error and Flow Control

Protocol 2: The Simplex Stop & Wait Protocol

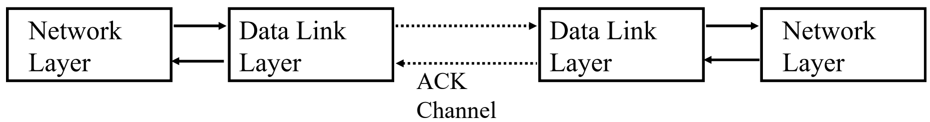
Assume Network
Layer always has
data to send



Error and Flow Control

Protocol 2: The Simplex Stop & Wait Protocol

Assume Network Layer always has data to send



Error and Flow Control

Protocol 3: The Simplex Protocol for a Noisy Channel

- Assumptions
  - One directional information flow
  - Network Layer always has a packet to send
  - Finite receive buffers
  - Allow errors or lost packets
- Data link protocols must address
  - **When to retransmit**
  - **What to retransmit**

Multiple ways of answering these questions; the answer differentiates DLC protocols

Error and Flow Control

Protocol 3: The Simplex Protocol for a Noisy Channel

- **Timeout** to determine when to retransmitt
- Example:
  - Assume a 1 ms propagation time
  - Assume a .1 ms receiver packet processing time
  - Timeout interval >2.1 ms
    - If no acknowledgment received in 2.1 ms then,
      - Packet in error
      - Acknowledgment lost

Error and Flow Control

Protocol 3: The Simplex Protocol for a Noisy Channel

- Timeout interval too short
  - Duplicate packets
- Timeout interval too long
  - Reduced throughput

Error and Flow Control

Protocol 3: The Simplex Protocol for a Noisy Channel

- **Sequence numbers** are used to determine what to retransmit
  - Transmitter assigns a number to each frame
  - Receiver keeps track of the expected frame number
  - How to deal with out of sequence frames, i.e., if the received sequence number does not *match* what is expected,
    - The frame is dumped (go-back-N)
    - Frame stored (Selective Repeat)

Error and Flow Control

Sliding Window Protocols: Assumptions

- Two directional information flow
- Network Layer always has a packet to send
- Finite receive buffers
- Finite number of bits/sequence number
- Bit errors
- Piggybacking
  - Put Acknowledgments in reverse traffic flow
  - Increases protocol efficiency
  - Reduces interrupts

Error and Flow Control

Sliding Window Protocols:

- Send more that one packet before receiving an ACK

  Advantage→ pipeline

- Why called sliding window
  - Assume 2 bits/Sequence number
  - Possible frame numbers 0, 1, 2, 3

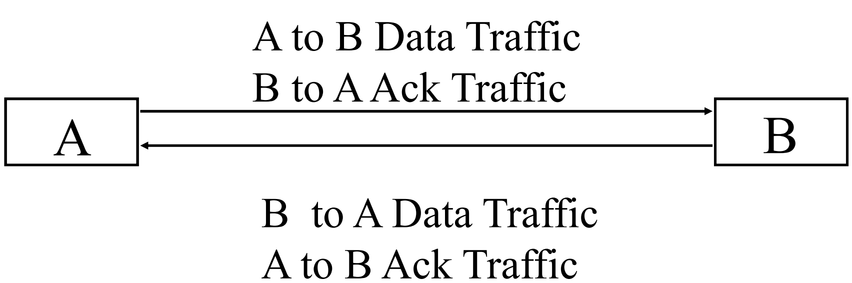
0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3

Receive ack and advance window

Design issue: how to set #bits/SN

Error and Flow Control

Sliding Window Protocols:



Error and Flow Control
Sliding Window Protocols

- Transmitter keeps a list of sequence #'s it can use
  - Sending window
- Receiver keeps a list of sequence #'s it will accept
  - Receiving window
- n = # bits/(sequence number)

Error and Flow Control
Sliding Window Protocols

- Sequence numbers in range 0...2^n-1
- This allows N=2^n-1 packets to be sent before getting and acknowledgment
- Requires N=2^n-1 packets buffers
  - Why not use all 2^n seq #'s, for n =3 then have 0…7 (8 seq #'s)

Error and Flow Control

Sliding Window Protocols: How many frames can be pipelined: Problem if max # frames in pipeline = 2^n

- Assume that # frames in pipeline $\leq 2^n$
- Assume n = 3, Node A sends 0...7 (8 frames)
- Node B receives 0...7 ok and sends Ack
- Now B expects next unique packet to have seq # = 0
- **First Ack gets lost**
- Packet 0 of Node A times out
- Node B receives another packet 0, expects a packet 0, but this is a duplicate
- Thus: # frames in pipline <= 2^n-1

Error and Flow Control

Sliding Window Protocols: How many frames can be pipelined (1)

- Now with # frames in pipline = N= 2^n-1
  - 0....6 (7 frames)
- Node A sends 0...6
- Node B receives 0...6 ok
- Node B sends Ack for packet 0
- Ack for packet 0 gets lost

Error and Flow Control

Sliding Window Protocols: How many frames can be pipelined (2)

- Node A times out
- Node A retransmits 0…6 (for go-back N)
- **But Node B is expecting frame #7**
- Node ignores 0…6 (often will send a RR frame explicitly telling Node A it is expecting Frame #7)

Error and Flow Control

- Types of sliding window protocols
  - Go-Back-N
  - Selective Repeat
- Focus on which frames to retransmit
- Pipeline: send up to N frames before receiving an acknowledgment
- Go-Back-N →Delete correctly received out of sequence frames
- Selective Repeat → Resend missing frame

Error and Flow Control

Performance Example

- Distance between nodes
  = 6600 km
- Frame length = 1000 bits
- Rate = 1.2Gb/s
- Large delay-bandwidth product network
  → $2\tau R$ = 52.8 Mb

Error and Flow Control

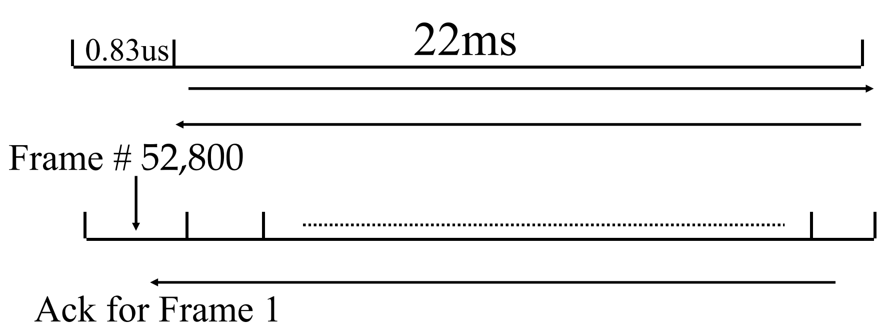
Performance Example

- Case 1: Stop and Wait (N=1)
  - Frame transmission time = 1000bits/1.2x10^9 b/s
    =0.83us
  - Propagation time = 6600x10^3 km/3x10^8m/s=22 ms
  - Transmit frame at t=0,
  - At 0.83us + 22 ms frame received
  - At 0.83us + 44ms the acknowledgment is received, therefore transmitted 1000 bits in (0.83us + 44ms)
  - Effective transmission rate is
    1000/44ms ~22.7kb/s
  - Efficiency:
    (22.7Kb/s)/(1.2Gb/s) ~ 0.002% efficient

Error and Flow Control
Performance Example

- Case 2:
  - $2\tau R/n_f$ = 26.4 Mb/1000=52,800
    (n = 16 # SN's = 2^{16} -1 ~64K)
  - Pipeline 52,800 frames,
  - Note with N=52,800 the first acknowledgment arrives at the transmitter just in time for the next frame to be transmitted. The transmitter is never blocked. The protocol is 100% efficient

Error and Flow Control
Performance Example



Error and Flow Control
Performance Example

- Distance between nodes = 1 km
- Frame length = 1000 bits
- Capacity = 150 Mb/s
- No errors
- Delay-bandwidth product
  - Assume free space
  - τ =1000m/c = 3.33 us → Access Network
  - 2 τR= 1000 bits (one frame in RTT)

Error and Flow Control
Performance Example

- Case 1: Stop and Wait (N=1)
  - Frame transmission time = 6.66us
  - Propagation time = 3.33us
  - Transmit frame at t=0,
  - At 6.66 us + 3.33us frame received
  - At 6.66us + 6.66us the acknowledgment is received, therefore transmitted 1000 bits in 6.66us + 6.66us
  - Effective transmission rate is 1000/13.3us ~ 75Mb/s
  - Efficiency: (75Mb/s)/(150Mb/s) ~ 50.0% efficient

Error and Flow Control

Performance Example

- Case 2: Stop and Wait (N=1)
  - **Reduce capacity** → 1.5 Mb/s
  - Frame transmission time = 666us
  - Propagation time = 3.33us
  - Transmit frame at t=0,
  - At 666 us + 3.33us frame received
  - At 666us + 6.66us the acknowledgment is received, therefore transmitted 1000 bits in 666us + 6.66us
  - Effective transmission rate is 1000/672us ~ 1.488 Mb/s
  - Efficiency: (1.488Mb/s)/(1.50Mb/s) ~ 99.2% efficient

Error and Flow Control

Performance Example

- Case 3: Stop and Wait (N=1)
  - Capacity to 150 Mb/s
  - Frame transmission time = 6.66us
  - **WAN** → D=1000km Propagation time = 3333us
  - $2\tau R$ = 1Mb → # frames in RTT = $2\tau R/n_f$ = 1000
  - Transmit frame at t=0,
  - At 6.66 us + 3333us frame received
  - At 6.66us + 6666us the acknowledgment is received, therefore transmitted 1000 bits in 6.66us + 6666us
  - Effective transmission rate is 1000/6672us ~ .149Mb/s
  - Efficiency: (.149Mb/s)/(150Mb/s) ~ 0.1% efficient

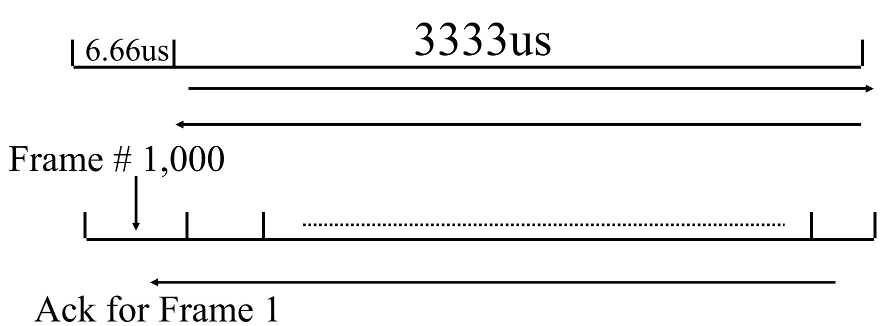
Error and Flow Control

Performance Example

- Case 4: Sliding window (N=1023; n=10 or 10bits/seq #)
  - Capacity to 150 Mb/s
  - Frame transmission time = 6.66us
  - WAN: D=1000km Propagation time = 3333us
  - Transmit frame at t=0,
  - Note 2 τR ~ 1Mb or in frames 1000 frames
  - Since time to transmit 1023 frames > 1000
    - Always have a sequence number to use
    - Never have to wait for ACK
  - Efficiency→ 100%

Error and Flow Control

Example



Error and Flow Control

Go-Back-N Protocol (1)

- Problem:

  If there is an error or lost frame then what rules are used to determine the frames to retransmit.

- Go-back-N
  - Retransmit all frames transmitted after the erred frame
  - The receiver ignores all out-of sequence frames, out-of sequence frames dropped

Error and Flow Control

Go-Back-N Protocol (2)

Example:

Transmit 1,2,3,4,5 and

frame 2 is in error then

3, 4, and 5 are received out of sequence and

retransmit 2,3,4,5

Error and Flow Control
Selective Repeat

- Receiver accepts out of sequence frames
- Requires buffers in receiver and transmitter
- Requires extra processing to deliver packets in order to the Network Layer

Animations of DLC

- http://www.ccs-labs.org/teaching/rn/animations/gbn_sr/
- https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/go-back-n-protocol/index.html
- https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/selective-repeat-protocol/index.html

Error and Flow Control

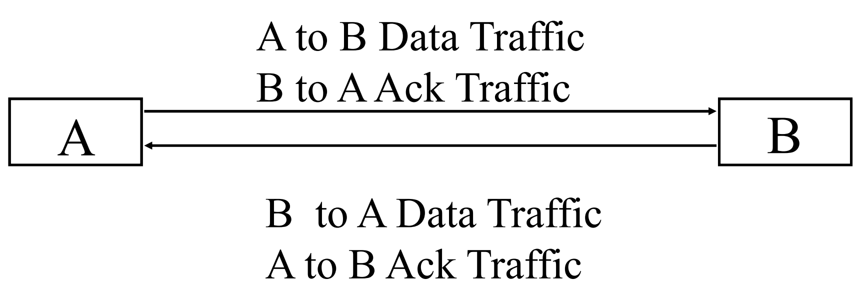
Other Enhancements

- Negative Acknowledgment
  - When an out-of-sequence frame is received the receiver sends a **NAK** frame to the transmitter, the **NAK** frame contains the sequence number of the expected data frame.
  - **NAK** enables faster error recovery, without a **NAK** time-out must be used to learn about errors.

Error and Flow Control

Sliding Window Protocols: Piggyback **ACKS**

Reverse traffic is used to Piggyback **ACKS**



Error and Flow Control

Other Enhancements: Acknowledgment timer

- If there is light (or no) reverse traffic then **ACKS** may not be sent.
- An acknowledgment timer is used to insure **ACKS** are sent.
- Upon receipt of a frame an *AckTimer* is started. If reverse traffic arrives before the *AckTimer* fires then piggyback the **ACK**. If the *AckTimer* fires then send a supervisory ACK frame.

Error and Flow Control

Performance

- Definition for effective rate

$$R_{eff} = \frac{\#\ \text{Bits Delivered}}{\text{Time to transfer Bits given the protocol}}$$

Error and Flow Control

Performance

- Length of data packet (bits) = D
- Number of overhead bits/packet = n_o
- Link Rate (b/s) = R
- Length of Ack Packet (bits)= n_a
- Frame size = n_f = D+ n_o
- One-way propagation delay = τ
- Processing time
  (in receiver and transmitter) = t_{proc}

Error and Flow Control

Performance-Stop & Wait

- Effective rate and efficiency for simplex stop-and-wait protocol
  - $t_f = n_f/R$
  - $t_{ack} = n_a/R$
  - Time to transmit one frame = t_o

    $t_o = 2\,\tau + t_f + t_{ack} + 2\,t_{proc} = 2(\tau + t_{proc}) + (n_a + n_f)/R$

Error and Flow Control

Performance-Stop & Wait

- $R_{eff} = (n_f - n_o) / t_o = D / t_o$
- $Efficiency = R_{eff}/R =$

$$\eta_o = \frac{1 - \frac{n_o}{n_f}}{1 + \frac{n_a}{n_f} + \frac{2R(\tau + t_{proc})}{n_f}}$$

Error and Flow Control

Performance-Stop & Wait: Limiting Case

Assuming

1) $n_a \langle\langle n_f$ so $\frac{n_a}{n_f} \longrightarrow 0$

2) $t_{proc} \langle\langle \tau$ so $t_{proc} + \tau \approx \tau$

3) $n_o \langle\langle n_f$ so $\frac{n_o}{n_f} \longrightarrow 0$

then

$$\eta_o = \frac{1}{1 + \frac{2\tau R}{n_f}}$$

$\frac{2\tau R}{n_f} =$ # frames in RTT

Define 2τR =
Delay-Bandwidth Product

For fixed DLL parameters
As Delay-Bandwidth Product ↑
Efficiency ↓

N_{RTT} = # Frames in RTT

$$\eta_o = \frac{1}{1 + N_{RTT}}$$

Error and Flow Control
Performance-Stop & Wait

- Example
  - Frame size = 1024 bytes
  - Overhead = Ack = 8 bytes
  - τ = 50 ms
    - Case 1: R=30 Kb/s →Efficiency = 73%
    - Case 2: R=1.5 Mb/s →Efficiency = 5%

Error and Flow Control
Performance-Sliding Window Protocol

- Case 1: Large window
  - Window Size = N = 2^n -1
  - Transmit N packet and wait for Ack
  - Making the same assumption as before
  - First Ack arrives at sender at:

$$2\tau + \frac{n_f}{R}$$

Error and Flow Control

Performance-Sliding Window Protocol

- Case 1: Large window
  - If time to transmit N packets > time to get first ack
    - Or $Nn_f/R > 2\tau + n_f/R$, or $N > 2\tau R/n_f + 1$
    - Then channel is always busy sending packets
    - Efficiency = $\eta \sim 1$ [if accounting for overhead then $\eta_o=(n_f - n_o)/n_f$]

Error and Flow Control

Performance-Sliding Window Protocol

- Case 2: Small Window
  - If time to transmit N packets < time to get first ack
    - Or $Nn_f/R < 2\tau + n_f/R$
    - Then channel is **Not** always busy sending packets: **Time is wasted waiting for an Ack**

Error and Flow Control

Performance-Sliding Window Protocol

- Time to send one window = Nn_f/R
- Number of bits sent = Nn_f
- Time to send Nn_f bits = $2\tau + n_f/R$
- Effective rate = $Nn_f/(2\tau + n_f/R)$
- Efficiency $= \eta_o$
  $= Nn_f/(2\tau R + n_f)$
  $= N/(1 + 2\tau R/n_f)$
  $= N/(1 + \#\text{ packets in RTT})$

Case 2: Small Window

If $Nn_f/R < 2\tau + n_f/R$ then

N_{RTT} = # Frames in RTT

$$\eta_o = \frac{N}{1+N_{RTT}}$$

Error and Flow Control

Performance-Sliding Window Protocol

- Example:
  - Frame size = 1024 bytes
  - Overhead = Ack = 0 bytes
  - τ = 1 ms
  - Rate = 40 Mb/s
    - Case 1: N = 12 → Efficiency = 100% → 40 Mb/s
    - Case 2: N = 8 → Efficiency = ~75% → 30 Mb/s
    - Case 3: N = 4 → Efficiency = ~ 37% →15 Mb/s

Note you can control the rate by changing N

Error and Flow Control

Performance-Stop & Wait with Errors

- Let p = Probability of a bit error
- Assume bits errors are random
- Let P_f = Probability of a frame error
- $P_f = 1 - (1-p)^{n_f}$
- If $p << 1$ then $P_f \sim p n_f$
- For stop & wait $R_{eff\text{-}with\ errors} = (1 - P_f)\, R_{eff}$

Open Loop Control

- Concept
  - Establish an expectation on the nature of the traffic generated by a source
    - Average rate
    - Maximum burst size, e.g., number of consecutive bits transmitted
  - If traffic exceed the expectation (traffic contract) then
    - Tag packet as discard eligible (DE)
    - Discard or loss probability
    - Possible actions
      - Drop immediately: prevent packet from entering the network
      - Allow into the network but drop if congestion

Open Loop Control: Frame Relay Networks

- Negotiated Traffic Parameters
  - Committed Information Rate in b/s (CIR)
  - Committed Burst Size in bits (B_c)
  - Excess Burst Size in bits (B_e)
  - Measurement Interval in sec
    $T = B_c / CIR$

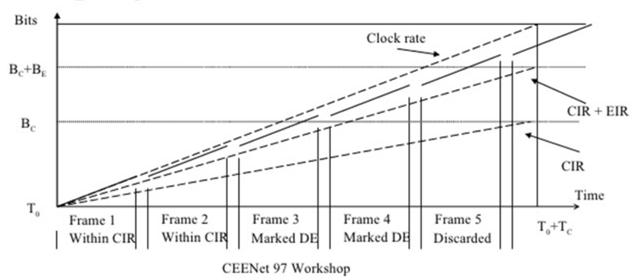
Open Loop Control: Frame Relay Networks

- Accept and "Guarantee" Delivery of Up To Bc in Any T (CIR in b/s)
- High Loss Priority (DE=0)
- Accept Up To (Bc + Be) More In Any T
- Low Loss Priority: Network May Discard If Congested (DE=1) EIR = Extended Information Rate (b/s)
- Excess Over (Bc + Be) in T Discarded At Access Point

Frame Relay

CIR and EIR - how does it work

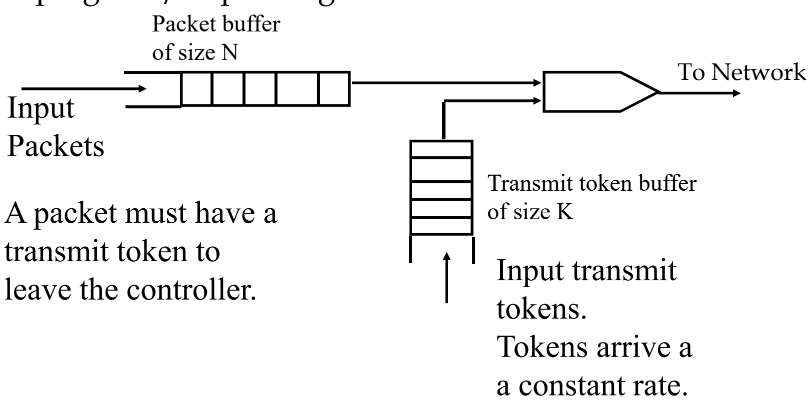
- $B_C = T_C * CIR$
- $B_E = T_C * EIR$



For more information see: ANSI T1S1/90-175R4, Addendum #1 (Congestion Management) to T1.606, 1990, p. 8.

Open Loop Control: Token Bucket Algorithm

- Open loop modification of the flow into the network.
- Traffic shaping and/or policing



A packet must have a transmit token to leave the controller.

Rate Control

Token Bucket Algorithm

- Modes of operation
  - Packets arriving to an empty token buffer are discarded when N=0.

  Or

  Packets arriving to an empty token buffer are marked when N>0
- Scheme controls
  - Average rate into the system
  - Maximum burst size into the system

Rate Control

Token Bucket Algorithm

- Example:
  - Suppose the system had no arrivals for a *long* time, then the packet buffer would be empty and the token buffer would be full, i.e. have K tokens.
  - A large burst of packets arrive.
  - K consecutive packets would be transmitted and then packets would be *leaked* into the systems at the token arrival rate.
- K controls the maximum burst size
- The token arrival rate controls the average transmission rate

Rate Control

Token Bucket Algorithm : Example

- Parameters
  - R = OC-12c = 622 Mb/s
  - Packet size 53 bytes
  - Token buffer holds 100 tokens
  - Inter-token time = 8.5 us.
- What is the average flow into the network in b/s?
  - 8.5us/token => 8.5us/packet
    117.6 x10^3 packets/sec →50 Mb/s
- What is the maximum burst size into the network?
  - 100 packets

Rate Control

Leaky Bucket Algorithm

- Leaky bucket algorithm is a special case of the token bucket.
- K =1 leaky bucket algorithm
- Maximum burst size = 1
- Both token and leaky bucket algorithms can work at byte or packet levels
- Violating packets can be either dropped or tagged
- Show Extend simulation

Data Link Control Standards

- HDLC
  - High level data link control
- LAPB
  - Link Access Protocol-Balanced
- LAPD
  - Link Access Protocol D
  - Used in ISDN and based on LAPB

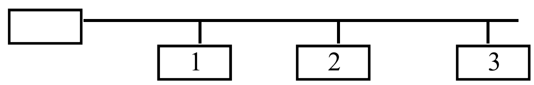
HDLC Frame types

- Information Frames (I-frames)
  - Carry user data
- Supervisory Frames (S-frames)
  - Carry control information
    - Acks
    - flow control
- Unnumbered Frames (U-frames)
  - Used for line initialization

Data Link Control Standards

Flag	Address	Control	Data	CRC	Flag
8 bits	8 bits	8 bits		16 bits	8 bits

• Address ⇒ Provide capability for multidrop lines

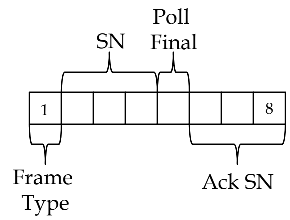


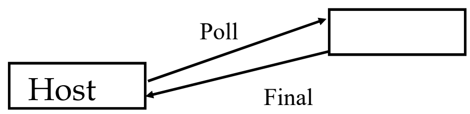
Data Link Control Standards

- Control
  - Sequence Numbers
  - Ack
  - Frame type
- Data
  - Network layer PDU
  - Variable length
- CRC

Data Link Control Standard

- Control structure I-frame
  - Bit 1 = 0 indicate I-frame
  - Bits 2-4 are the sequence number
  - Bit 5 is the Poll/Final (P/F) bit.
  - Bits 6-8 are the Next bits, i.e, sequence number for the piggyback ack.





Data Link Control Standard

- Control structure S-frames
  - Type 1: Receive Ready (RR)
    - Used to ack when no piggyback used
  - Type 2: Receiver-not-Ready (RNR)
    - Used to tell transmitter to stop sending
  - Type 3: Selective Repeat
    - Not used in LAPB and LABD

Data Link Control Standard

- Data link control protocol modes
  - Normal response mode (NRM)
    - –Master/slave
  - Asynchronous balanced mode (ABM)
    - –Equal partners

PPP:
The Internet Point-to-Point Protocol

- PPP is a variation of HDLC originally designed to encapsulate IP (and other) datagrams on dial-up or leased carrier circuits. PPP is used in "Packet over SONET" for high speed Internet connections

Flag 01111110	Address 1111111	Control 00000011	Protocol	Information	CRC	flag 01111110

Address: All stations are to accept the frame

Control: Unnumbered frame

Protocol: Specifies what kind of packet is contained in the payload, e.g., IP, IPX

PPP Frame Format

Summary

- Operation of DLC protocols
  - Frame structure
  - Go-back-N (N=1 is the Stop and Wait protocol)
  - Selective Repeat
  - Efficiency of DLC protocols
  - Standard DLC protocols → HDLC
- Open loop flow control