Data Link Control (DLC)

#7

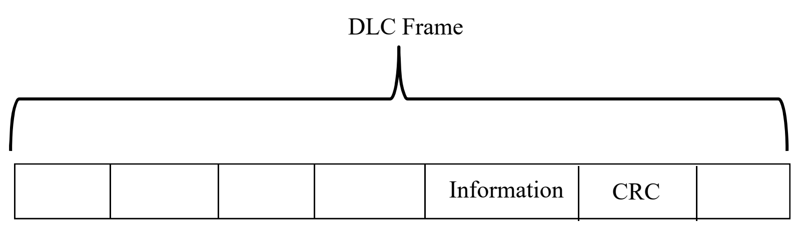
Outline

- Data link Control (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 Detection & Control
  - Flow Control

Building up the frame structure



CRC used to check for bit errors

Framing

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

Data to DLC layer from upper layer
0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

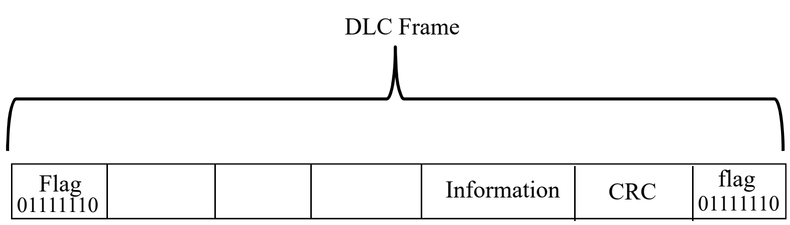
After bit stuffing
01111110 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0 01111110
Data to PHY Layer

Stuffed bits

After bit unstuffing
Data to DLC layer to upper layer
0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

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

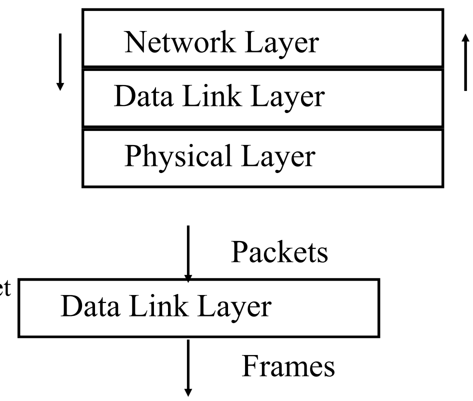
Building up the frame structure



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

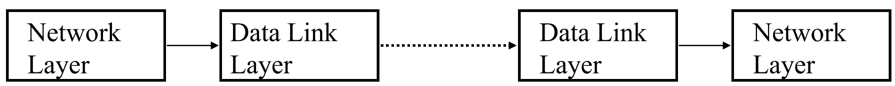
Events

- 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 (information in the header)
  - 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

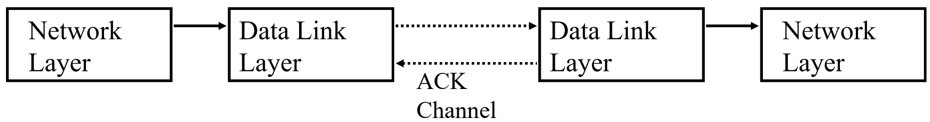
Protocol 2: The Simplex Stop & Wait Protocol: Assumptions

- 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

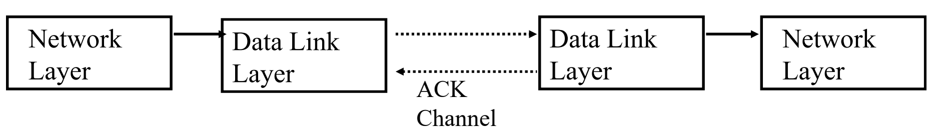
Assume Network Layer always has data to send



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

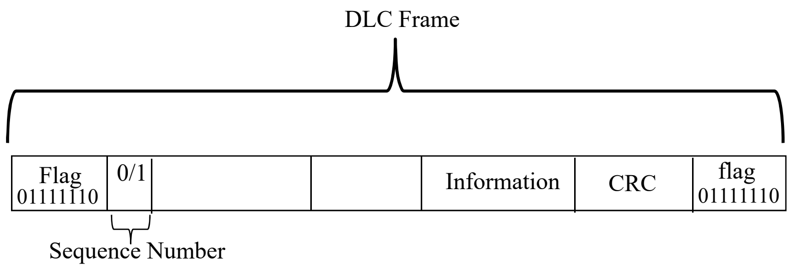
Protocol 3: The Simplex Protocol for a Noisy Channel

- One way to determine when to retransmit is with a **Timeout**
  - **Timeout** to used to trigger retransmission
- Example:
  - Assume a 1 ms propagation time
  - Assume zero clocking time (Packet Length)/R<<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 (packet correctly received)

Error and Flow Control
Protocol 3: The Simplex Protocol for a Noisy Channel

- Problem: Ack is dropped. Timer fires and source retransmits the packet. The destination receives duplicate packet. How does the destination know that it is a duplicate?
- Solution: Assign the packet a number 0 or 1.
  - Receiver keeps the "number of the expected packet"
  - If receives packet with "1" but expecting "0" then send new Ack for "1" telling sender that "1 was received"
  - The number is called a "sequence number" here number of bits/sequence number = 1 and number of packets the sender can send is 1= 2^1-1 (nore later)

Building up the frame structure



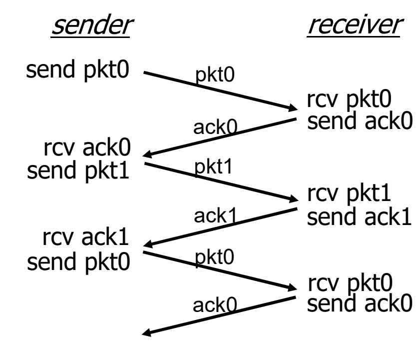
Protocol 3: The Simplex Protocol for a Noisy Channel

Approach: sender waits "reasonable" amount of time for ACK

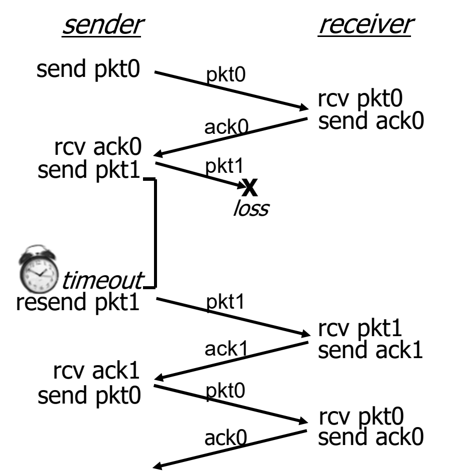
- retransmits if no ACK received in this time
- if pkt (or ACK) just delayed (not lost):
  - retransmission will be duplicate, seq #s will handle this!
  - receiver must specify seq # of packet being ACKed
- use countdown timer to interrupt after "reasonable" amount of time

Modified from: 8th edition Jim Kurose, Keith Ross Pearson, 2020

Protocol 3: The Simplex Protocol for a Noisy Channel in action

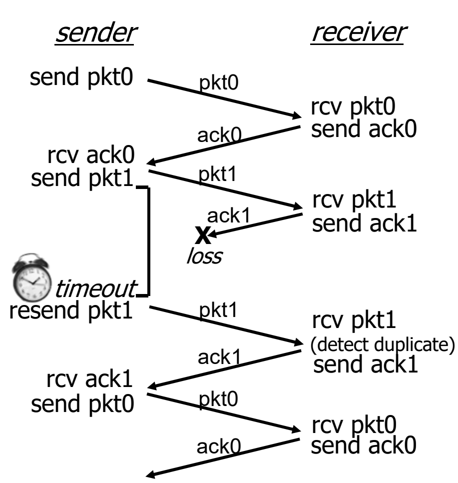


(a) no loss

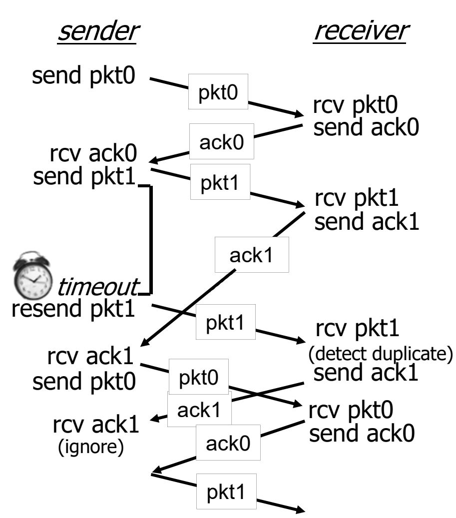


(b) packet loss

Protocol 3: The Simplex Protocol for a Noisy Channel in action



(c) ACK loss



(d) premature timeout/ delayed ACK

Error and Flow Control

Protocol 3: The Simplex Protocol for a Noisy Channel

- Another way to determine when to retransmit is with a **Negative Acknowledgement (NAK)**
- Example:
  - Receive Frame
  - Calculate checksum
  - Checksum not equal 0 then Frame in error
  - Receiver sends a **NAK** Frame back to the sender
  - Sender receives **NAK** and retransmits the Frame
- Using NAKs are often more efficient (faster) than timeout alone.
- Note will always need timeout method too, as NAKs can be 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

Performance Example

- Distance between nodes
  = 6600 km (like a WAN)
- 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
    =0.83μs
  - Propagation time = 6600x10^3 km/3x10^8m/s=22 ms
  - Transmit frame at t=0,
  - At 0.83μs + 22 ms frame received
  - At 0.83μs + 44ms the acknowledgment is received, therefore transmitted 1000 bits in (0.83μs + 44ms)
  - Effective transmission rate is
    1000/44ms ~22.7kb/s
  - Efficiency:
    (22.7Kb/s)/(1.2Gb/s) ~ 0.002% efficient

Pipelined protocols operation

pipelining: sender allows multiple, "in-flight", yet-to-be-acknowledged packets

- Send up to N packets before needed to get an Ack, N=Window size
- range of sequence numbers must be increased
- buffering at sender and/or receiver

(a) a stop-and-wait protocol in operation

Error and Flow Control

Protocol 3: The Simplex Protocol for a Noisy Channel

- **Sequence numbers** are used to determine <u>what</u> to retransmit
  - Sender assigns a number to each frame
  - Sender stores transmitted frames and keeps track of their sequence numbers.
  - Different protocols define which frames are retransmitted
  - 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)

Building up the frame structure

Window size=N= 2^n-1
n bits

Flag 01111110	Sequence #	Other	Other	Information	CRC	flag 01111110

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
  - Sequence number wrap around
  - Example: 2 bits/sequence
    - 00, 01,10, 11, then need to use 00 again
- 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 n=2 bits/Sequence number
  - Window size= N= 2^n-1 = 3
  - Possible frame numbers 0, 1, 2, 3

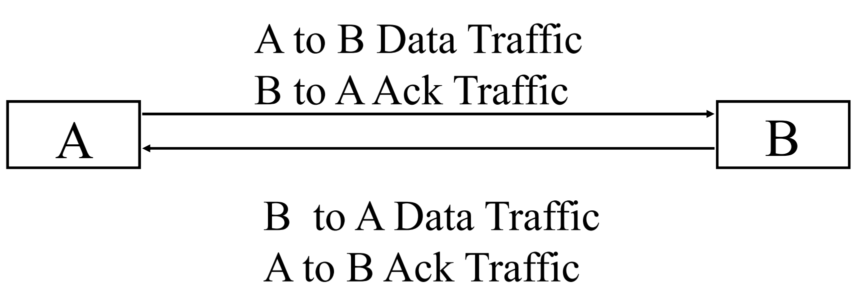
0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3

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

- Sender 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)

  n=number of bits in the packet header used for the sequence number
- Window size= N= 2^n-1

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 ≤ 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 Receive Ready (RR) frame explicitly telling Node A it is expecting Frame #7), e.g., using a NAK containing the expected frame #.

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 just the missing frame

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

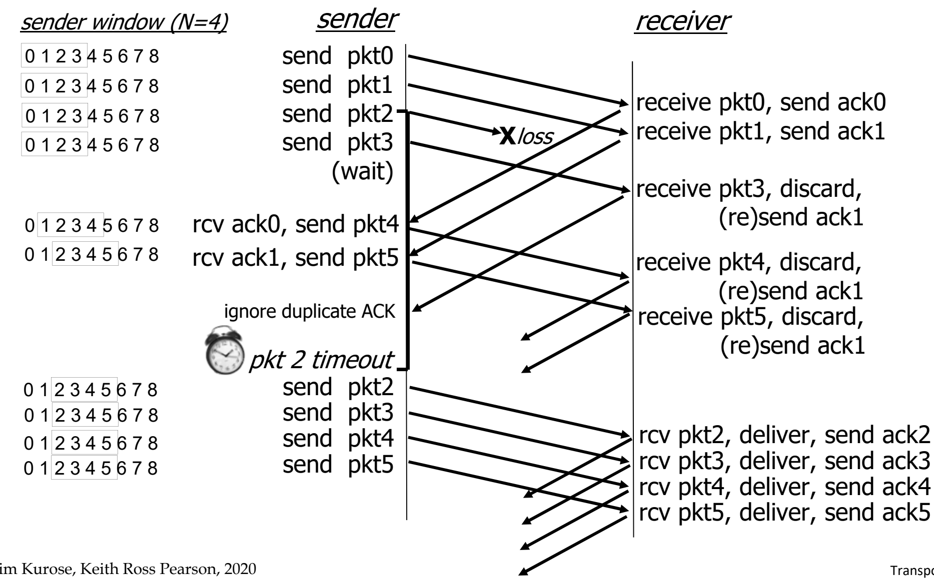
retransmit 2,3,4,5

Error and Flow Control

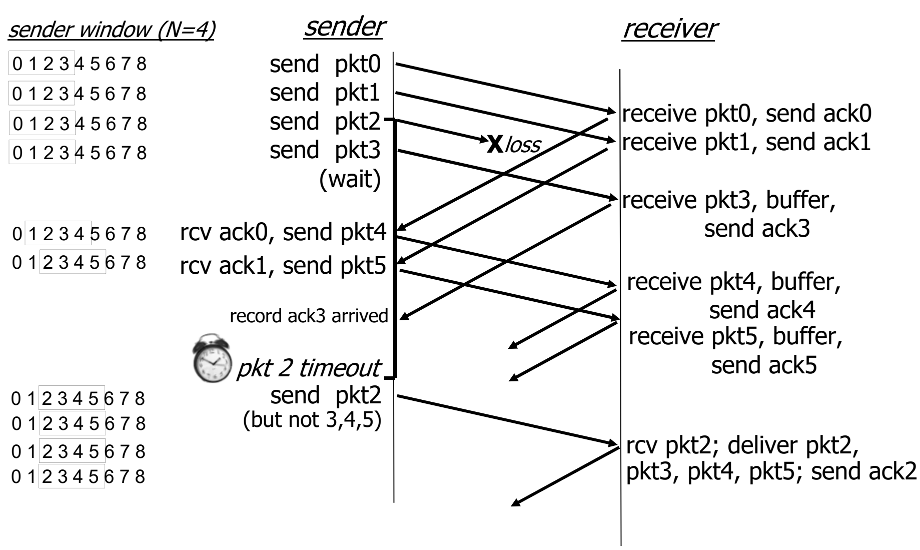
Selective Repeat

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

Go-Back-N in action



Selective Repeat in action



Error and Flow Control

Performance Example

- Distance between nodes = 6600 km τ=22ms
- Frame length = 1000 bits
- Rate = 1.2Gb/s
- Large delay-bandwidth product network

→τR = 26.4 Mb

→2τR = 52.8 Mb

  - Number of frames in RTT =2τR/n_f

    = 52.8 Mb/1000=52,800 (n = 16 so N= 2^{16}-1 ~64K)
  - Pipeline 52,800 frames,
  - Note with N=52,801 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

- Case 1: Stop and Wait (N=1)
  - Rate = 150 Mb/s
  - Distance between nodes = 1 km (like a LAN)
  - Frame length = 1000 bits
  - No errors
  - Delay-bandwidth product
    - Assume free space
    - τ =1000m/c = 3.33 μs

      → Access Network
    - 2 τR= 1000 bits (one frame in RTT)
  - Frame transmission time = 6.66μs
  - Propagation time = 3.33μs
  - Transmit frame at t=0,
  - At 6.66 μs + 3.33μs frame received
  - At 6.66μs + 6.66μs the acknowledgment is received, therefore transmitted 1000 bits in 6.66μs + 6.66μs =13.3μs
  - Effective transmission rate is 1000/13.3μs ~ 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 = 666μs
  - Propagation time = 3.33μs
  - Transmit frame at t=0,
  - At 666 μs + 3.33μs frame received
  - At 666μs + 6.66μs the acknowledgment is received, therefore transmitted 1000 bits in 666μs + 6.66μs
  - Effective transmission rate is
    1000/672μs ~ 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 = 6.66μs
  - **WAN** → D=1000km Propagation time = 3333μs
  - 2τR = 1Mb → # frames in RTT = $2\tau R/n_f$ = 1000
  - Transmit frame at t=0,
  - At 6.66 μs + 3333μs frame received
  - At 6.66μs + 6666μs the acknowledgment is received, therefore transmitted 1000 bits in 6.66μs + 6666μs
  - Effective transmission rate is
    1000/6672μs ~ .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 = 6.66μs
  - WAN: D=1000km Propagation time = 3333μs
  - 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
Go-Back-N Protocol (1)

- Problem:

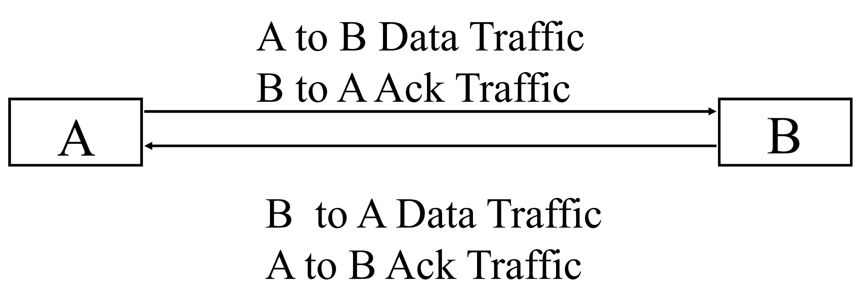
  If there is an error or lost frame then what rules are used to determine the frames to retransmit.
- Go-back-N
  - Retransmit all frames transmitted after the erred frame
  - The receiver ignores all out-of sequence frames, out-of sequence frames dropped
- Animations of DLC protocols
  - http://www.ccs-labs.org/teaching/rn/animations/gbn_sr/

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.
  - Countdown timer is always required in case the NAK is lost.

Error and Flow Control
Sliding Window Protocols: Piggyback **ACKS**

Reverse traffic is used to Piggyback **ACKS**



Building up the frame structure

Flag 01111110	Sequence # (n bits)	Piggyback Sequence # (n bits)	Other	Information	CRC	flag 01111110

Error and Flow Control

Other Enhancements: Acknowledgment timer

- If there is light (or no) reverse traffic then piggyback **ACKs** may not be sent.
- Solution:
  - 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 an explicit (or supervisory) ACK frame.

Table 3.1

Summary of reliable data transfer mechanisms and their use

Mechanism	Use, Comments
Checksum	Used to detect bit errors in a transmitted packet.
Timer	Used to timeout/retransmit a packet, possibly because the packet (or its ACK) was lost within the channel. Because timeouts can occur when a packet is delayed but not lost (premature timeout), or when a packet has been received by the receiver but the receiver-to-sender ACK has been lost, duplicate copies of a packet may be received by a receiver.
Sequence number	Used for sequential numbering of packets of data flowing from sender to receiver. Gaps in the sequence numbers of received packets allow the receiver to detect a lost packet. Packets with duplicate sequence numbers allow the receiver to detect duplicate copies of a packet.
Acknowledgment	Used by the receiver to tell the sender that a packet or set of packets has been received correctly. Acknowledgments will typically carry the sequence number of the packet or packets being acknowledged. Acknowledgments may be individual or cumulative, depending on the protocol.
Negative acknowledgment	Used by the receiver to tell the sender that a packet has not been received correctly. Negative acknowledgments will typically carry the sequence number of the packet that was not received correctly.
Window, pipelining	The sender may be restricted to sending only packets with sequence numbers that fall within a given range. By allowing multiple packets to be transmitted but not yet acknowledged, sender utilization can be increased over a stop-and-wait mode of operation. We'll see shortly that the window size may be set on the basis of the receiver's ability to receive and buffer messages, or the level of congestion in the network, or both.

Modified from: 8th edition Jim Kurose, Keith Ross Pearson, 2020

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 (bits) = n_f = D+ n_o
- One-way propagation delay (sec) = τ
- Processing time (sec)
  (in receiver and transmitter) = t_{proc}

Error and Flow Control
Performance-Stop & Wait

- Effective rate and efficiency for simplex stop-and-wait protocol
  - $t_f = n_f/R$ (clocking time of DLC frame)
  - $t_{ack} = n_a/R$ (clocking time of ACK packet)
  - Time to transmit one frame = t_o

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

Error and Flow Control

Performance-Stop & Wait

- $R_{eff} = (n_f - n_o)/t_o = D/t_o$
- $Efficiency = R_{eff}/R = \eta_0$

$$\eta_o = \frac{1 - \frac{n_0}{n_f}}{1 + \frac{n_a}{n_f} + \frac{2R(\tau + t_{proc})}{n_f}}$$

Error and Flow Control

Performance-Stop & Wait: Limiting Case

Assuming
(valid for today's networks)

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

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

Define 2τR =Delay-Bandwidth Product

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

$N_{RTT} = \frac{2\tau R}{n_f} = \#$ frames in RTT → $\eta_o = \frac{1}{1 + N_{RTT}}$

Interactive graphical tool→ **Stop & Wait Efficiency Trade-offs**

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
    (n bits in header for sequence number)
  - Transmit N packets 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 $N > 2\tau R/n_f + 1$
    - $N > 2\tau R/n_f + 1$ = <u>number packets in RTT + 1</u>
    - Then channel is always busy sending packets
    - Efficiency = $\eta \sim 1$
    - When 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 **<u>Not</u>** always busy sending packets:
      <u>Time is wasted waiting for an Ack</u>

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} = # Packets in RTT

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

Interactive graphical tool→ **Sliding Window Efficiency Trade-offs**

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

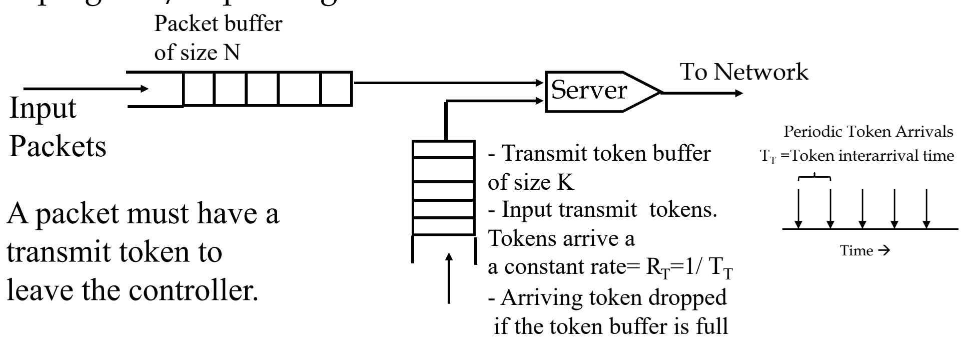
  (statistically independent)
- Let P_f = Probability of a frame error
- $P_f = 1 - (1-p)^{n_f}$
- If $p \ll 1$ then $P_f \sim p n_f$
- For stop & wait $R_{eff\text{-}with\ errors} = (1 - P_f)\, R_{eff}$

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)
    - Possible actions
      - Drop immediately: prevent packet from entering the network
      - Allow into the network but drop if congestion
  - Traffic control occurs at output port of router/switches

Open Loop Control: Token Bucket Algorithm

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



Rate Control
Token Bucket Algorithm

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

  Or

  Packets arriving to an empty token buffer are marked (DE) when packet buffer is not empty, N>0
- Scheme controls
  - Average rate into the system
  - Maximum burst size into the system

Rate Control

Token Bucket Algorithm

- Operation:
  - 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 = 100 Mb/s
  - Packet size 1000 bytes
  - Token buffer holds 100 tokens
  - Inter-token time = 20 µs.
- What is the average flow (in Mb/s) into the network in b/s?
  - 20 µs/token => 20 µs/packet
    60000packets/sec → (60000 packets/sec)(8000bits/packet)= 400 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)

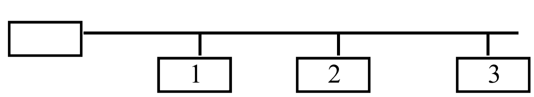
HDLC Frame types

- 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

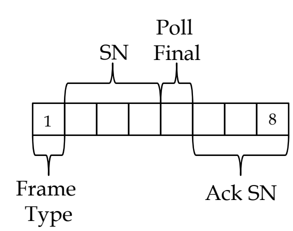


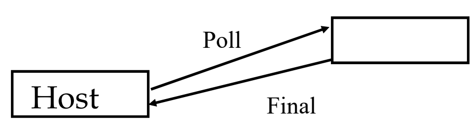
Data Link Control Standards

- 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)
    - Leader/Follower
  - 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	Address	Control	Protocol	Information	CRC	flag
01111110	1111111	00000011				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

Extra Slides

Open Loop Control

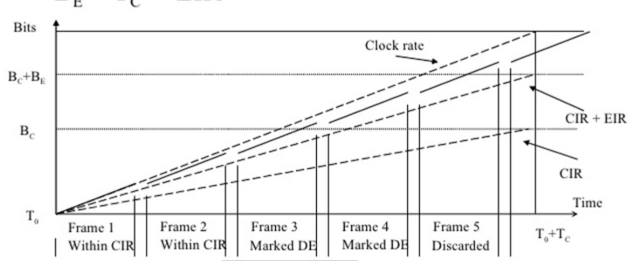
- Negotiated Traffic Parameters
  - Committed Information Rate in b/s (CIR)
  - Committed Burst Size in bits (B_c)
  - Excess Burst Size in bits (B_e)
  - Measurement Interval in sec
    $T = B_c / CIR$ ($CIR = B_c / T$)

Open Loop Control

- 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 Frame
  - Discard eligible-DE set DE bit =1, if "Congested" drop frames with DE=1
  - EIR = Extended Information Rate (b/s)
- Excess Over (Bc + Be) in T Discarded At Access Point

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.