ECE 356/COMPSI 356
Computer Network Architecture

Link Layer

Wednesday September 4th, 2019

Duke UNIVERSITY

Recap

- Last lecture:
  - Network programming
  - Physical layer
- Readings for this lecture: **PD 2.2 – 2.5**

2

Duke UNIVERSITY

Quiz 1 Review

- Average 8.46/10, median 9/10
- Questions with most incorrect answers
  - Delay-bandwidth product calculations: additional practice in the first homework
  - Internet history

History of Internet Service Providers (1/2)

- Internet started as a government-supported restricted-use tool (ARPANet 1969, NSFNet 1985-1995)
- 1980s: limited capabilities offered by commercial providers
  - E.g., e-mail
- First commercial Internet Service Providers (ISPs) appeared in 1989
  - Internet available to consumers
  - → Internet as a consumer phenomenon is approximately 30 years old

History of Internet Service Providers (2/2)

- 10,000 ISPs by late 1990s
- AOL leading provider by year 2000
- Will talk more about ISPs in a lecture on Inter-domain routing

Web Browsers

- Application for accessing information on the World Wide Web
  - Retrieve information from a web server and display it on user's device
- Browsers differ in features, latency of various tasks

History of Web Browsers (1/4)

- First web browser: 1991
- First popular web browser: Mosaic 1993
  - Hugely influential
  - Leader of Mosaic team created Netscape Navigator

ANDREESSEN
HOROWITZ

History of Web Browsers (2/4)

- Different browsers captured almost the entire market at different times
- https://www.reddit.com/r/dataisbeautiful/comments/cxuah9/usage_share_of_internet_browsers_1996_2019_oc/

History of Web Browsers (3/4)

- Netscape dominated the market in 1995
- Then: Internet Explorer (IE), first available to all Windows users, then integrated with Windows
  - ➢ Intense competition
  - ➢ Netscape: small one-product company, IE part of Microsoft ecosystem
  - ➢ Many people purchasing computers for the first time: no incentives to try non-default browsers
- Peak market share of Internet Explorer: **96%**

9

Duke UNIVERSITY

History of Web Browsers (4/4)

- Apple effect: IE discontinued on iOS platform → helped Safari, an Apple product
- Dominance of Android devices: Chrome pre-installed on Android
- Arguably, quality is a factor
- "*Chrome won the second browser war*"
  - ➢ Market share 60%

10

Duke UNIVERSITY

Link Capacity: Shannon-Hartley Theorem Example (1/2)

- $C = B*log_2(1+S/N)$
  - C in bps, B in Hz, S avg. signal power, N avg. noise power
- $SNR\ in\ dB = 10*log_{10}(S/N)$
- Example: B = 100,000 Hz, SNR = 3dB. What is the upper bound on link capacity?
- Calculating S/N: $3 = 10*log_{10}(S/N) \rightarrow 0.3 = log_{10}(S/N) \rightarrow$ $\mathbf{S/N = 2}$
- $C = 100{,}000*log_2(3) = 100{,}000*1.58 = 158{,}000\ bps$

Duke UNIVERSITY

Link Capacity: Shannon-Hartley Theorem Example (2/2)

- What is the increase in the upper bound of link capacity if we double the bandwidth?
  - It doubles (158,000*2 = 316,000 bps)
- What is the increase in the upper bound of link capacity if we double the SNR?
  - S/N = 4, B = 100,000
  - C = 100,000*log2(5) = 100,000*2.32 = 232,000

12

Duke UNIVERSITY

Shannon Theorem: Intuition

- Increase in B: more information can be sent
- Increase in SNR: fewer bit errors, more information can be received correctly

13

Duke UNIVERSITY

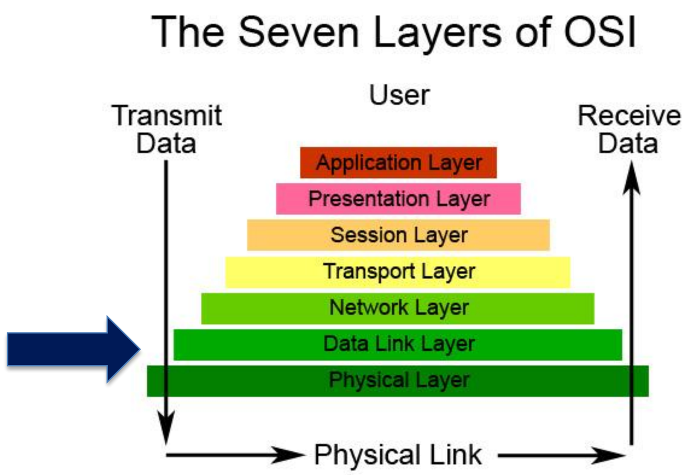
Lecture Outline

- Link layer: an introduction
- Encoding
- Framing
- Error detection
  - Parity, checksum, CRC
- Reliability
  - FEC, sliding window

14

Duke UNIVERSITY

Link Layer: an Introduction

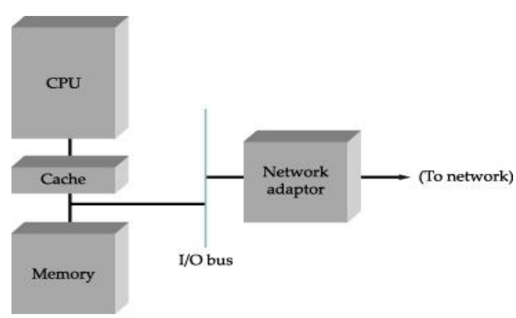


15

Duke UNIVERSITY

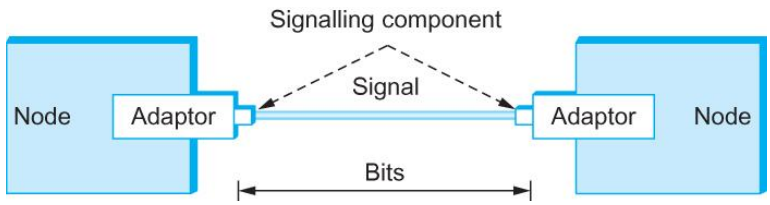
Recap: Put Bits on the Wire (1/2)

- Each node (e.g. a PC) connects to a network via a **network adaptor**
- The adaptor delivers data between a node's memory and the network
- A device driver is the program running inside the node that manages the above task



Duke UNIVERSITY

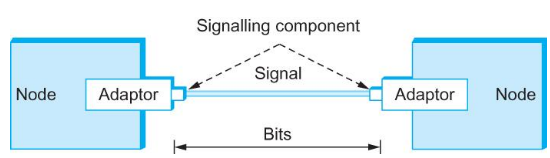
Recap: Put Bits on the Wire (2/2)



- At one end, a network adaptor encodes and modulates a bit into signals on a physical link
- At the other end, a network adaptor reads the signals on a physical link and converts it back to a bit

Duke UNIVERSITY

Metrics to Describe a Link



- Bandwidth
  - ➢ Why are some links slow/fast?
- Latency/delay
- Transmission delay (serialization)
  - ➢ Store and forward
- Delay-bandwidth product
- Throughput
  - ➢ How long does it take to send a file?

Duke UNIVERSITY

Link Layer Functions

- Encoding
- Framing
- Error detection
- Reliable transmission
- Managing multiple access (next week)

Duke UNIVERSITY

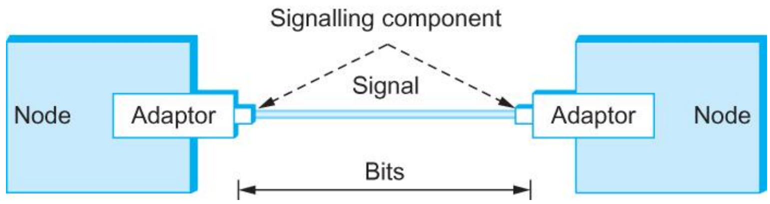
Lecture Outline

- Link layer: an introduction
- **Encoding:** NRZ, NRZI, Manchester coding, 4B/5B
- Framing
- Error detection
  - Parity, checksum, CRC
- Reliability
  - FEC, sliding window

20

Duke UNIVERSITY

Encoding

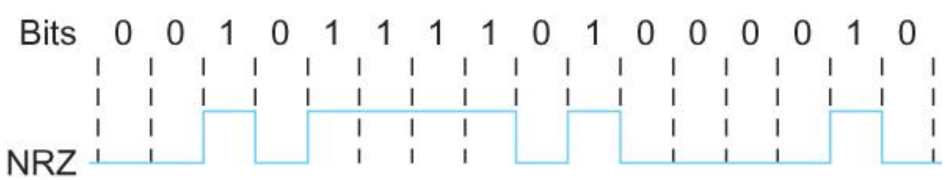


- Implemented in hardware
- High and low signals, ignore modulation
- Simplest one: 1 to high, 0 to low

Duke UNIVERSITY

Non-return to Zero (NRZ)

- 1 to high, 0 to low

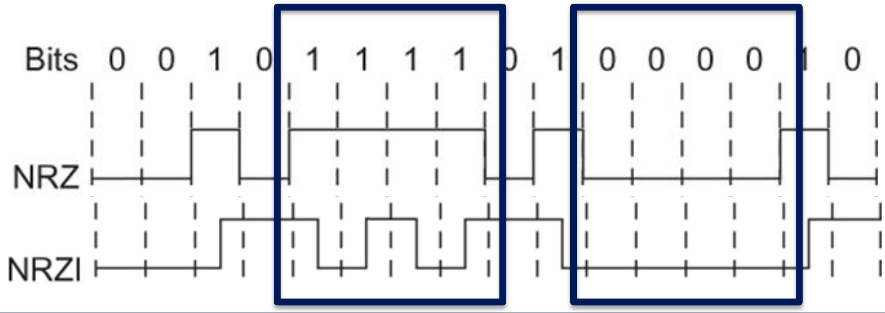


- Not good for decoding
  - ➢ Baseline wander for long sequences of 1s or 0s
  - ➢ Clock recovery difficulties

Duke UNIVERSITY

Non-return to Zero Inverted (NRZI)

- A transition from current signal encodes 1
- No transition encodes 0
- Does it solve all problems?
  - ➢ Not for consecutive 0s



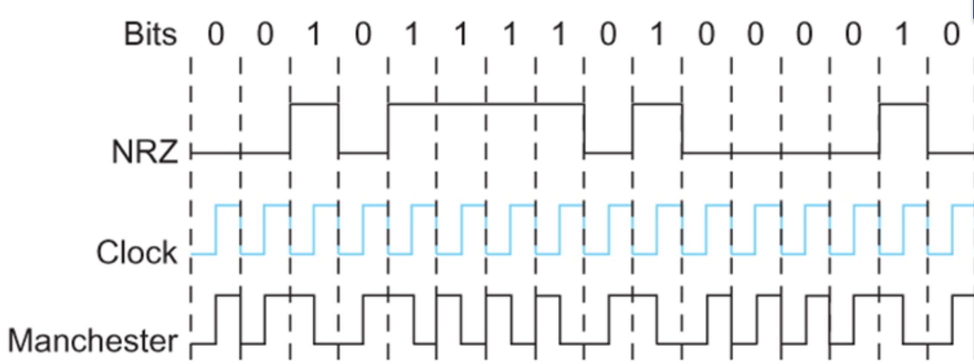
Duke UNIVERSITY

Manchester Encoding

- Transmit XOR of NRZ data and the clock
  - ➢ 1: high → low; 0: low → high

Inputs		Outputs
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

XOR truth table



Duke UNIVERSITY

Manchester Encoding Drawback

- Doubles the rate at which signals are sent
- Baud rate: signal change rate
- For Manchester encoding:
  - Bit rate is half of baud rate → only 50% efficient
  - If the receiver was able to keep up with the faster baud rate, NRZ and NRZI could have been able to transmit twice as many bits in the same time period

4B/5B

- Key idea: insert extra bits to break up long sequences of 0s or 1s
- 4-bit of data (“4B”) are encoded in a 5-bit code word (“5B”)
  - 16 data symbols, 32 code words
  - At most one leading 0, at most two trailing 0s
  - When sent back to back: no more than three consecutive 0s
- 5-bit codes are sent using NRZI
  - NRZI: transition encodes a 1, no transition encodes 0
  - Solves NRZI’s problem of consecutive zeroes
- 80% efficient

4-bit data symbol	5-bit code
0000	11110
0001	01001
0010	10100
0011	10101
0100	01010
0101	01011
0110	01110
0111	01111
1000	10010
1001	10011

4-bit data symbol	5-bit code
1010	10110
1011	10111
1100	11010
1101	11011
1110	11100
1111	11101

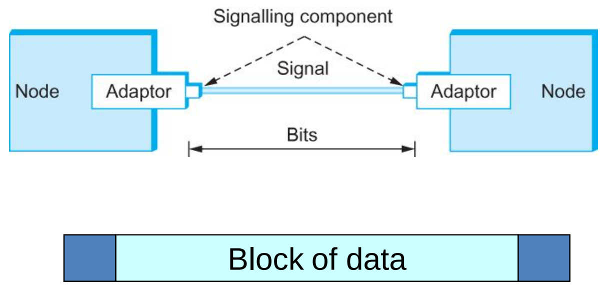
- Exercise:
  - 00101101
- What's the high/low signal sequence?

Lecture Outline

- Link layer: an introduction
- Encoding
- **Framing**
- Error detection
  - Parity, checksum, CRC
- Reliability
  - FEC, sliding window

Framing

- Now we've seen how to encode bitstreams
- But nodes send blocks of data (frames)
  - A's memory → adaptor → adaptor → B's memory
- An adaptor must determine the boundary of frames

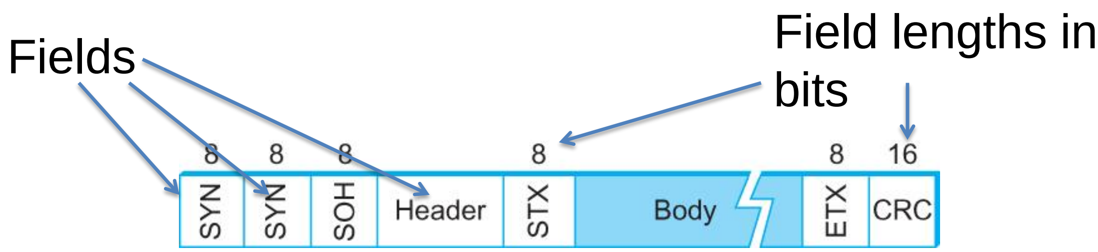


Variety of Framing Protocols

- Byte-oriented protocols
  - Sentinel approach
  - Byte-counting approach
  - Binary Synchronous Communication (BISYNC), Point-to-Point Protocol (PPP), DDCMP
- Bit-oriented protocols
- Clock-based framing

Convention: Illustrating Frame and Packet Formats

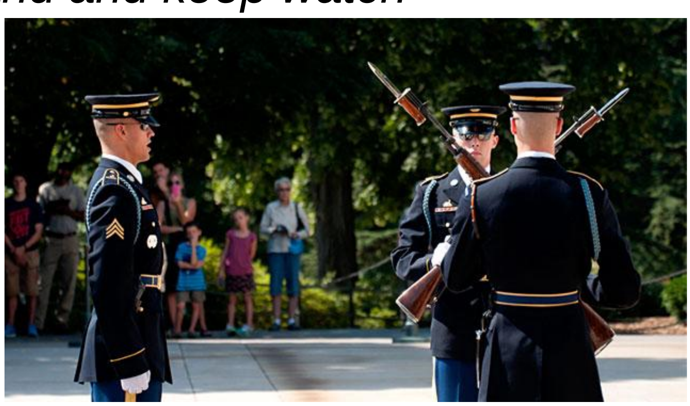
- Frame: a collection of bytes (characters)



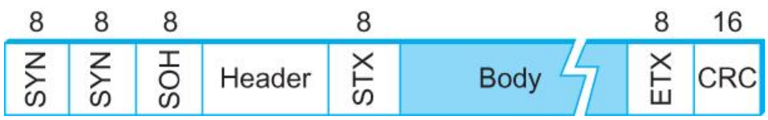
- Transmitted from the leftmost bit
  - ➢ Displayed start to end

Sentinel (n)

- Oxford dictionary: *A soldier or guard whose job is to stand and keep watch*



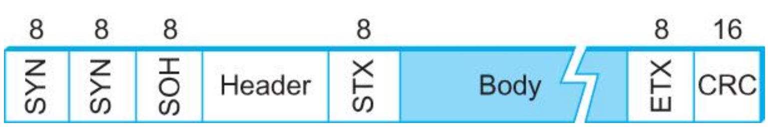
Byte-oriented Protocols: Sentinel Approach (1/2)



- Binary Synchronous Communication (BISYNC) by IBM in late 60s

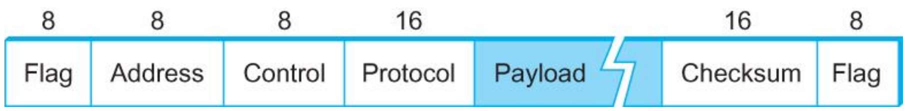
- Sentinel characters
  - Synchronization: SYN
  - Start of header: SOH
  - Start of text: STX, end of text: ETX
- Cyclic redundancy check (CRC): will discuss later

Byte-oriented Protocols: Sentinel Approach (2/2)



- What if special characters appear in a data stream?
  - "Escape character": DLE (data-link escape) preceding ETX when it appears in text
  - DLE preceded with another DLE
  - *Character stuffing*
    - Size of what goes on the wire is content-dependent

Point-to-Point Protocol (PPP)



- Commonly run over Internet links
- Sentinel approach: special *start of text character* denoted as Flag
  - 0 1 1 1 1 1 1 0
- Address, control : default numbers
- Protocol for demux : IP / IPX
- Payload : negotiated (1500 bytes)

Byte-oriented Protocols: Byte Counting Approach

- Digital Data Communication Message Protocol (DDCMP) by DECNET

- A byte count field: specify how long the frame is
- The corruption of the count field: *framing error*
  - Will count what is received up to the faulty count value as frame bits
  - The sentinel approach: corrupted ETX

Bit-oriented Protocols: HLDC

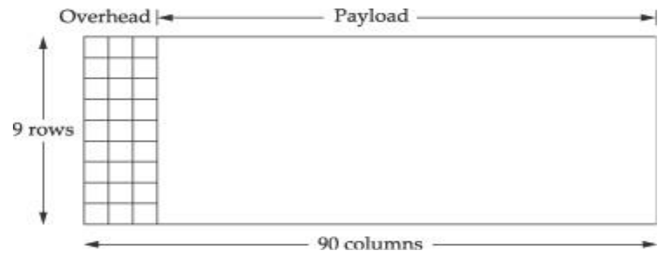
• High-level data link control (HDLC) protocol

- Frame: a collection of bits
- Beginning/ending sequence: 0111 1110
- Transmitted when a link is idle, to help devices synchronize clocks:
  - 0111 1110 or idle flags 1111 1111
- Bit-stuffing for data
- Frames are of variable length, their size *content-dependent*

HLDC Bit-stuffing Algorithm

- Any time five consecutive 1's have been transmitted from the body of the message (i.e. excluding when the sender is trying to send the distinguished 0111 1110 sequence)
  - The sender inserts 0 before transmitting the next bit

Clock-based Framing



- Synchronous Optical Network (SONET)
  - Dominant standard for optical networks
- Each frame is 125 μs long
  - 810 bytes for the slowest SONET link
- Clock synchronization
  - Special pattern repeated enough times

Duke UNIVERSITY

Lecture Outline

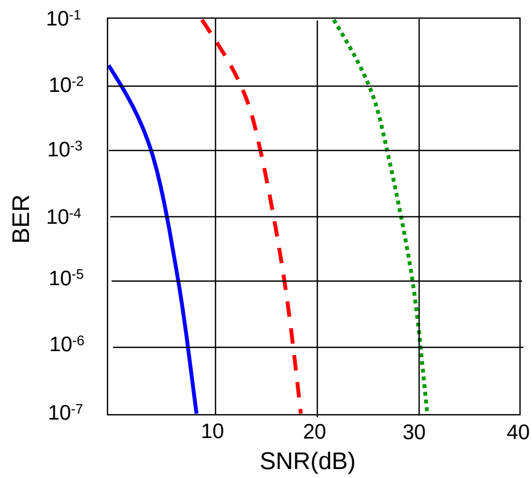
- Link layer: an introduction
- Encoding
- Framing
- **Error detection**
  - **Parity, checksum, CRC**
- Reliability
  - FEC, sliding window

40

Duke UNIVERSITY

Errors on Links

- Causes:
  - ➢ Noise, interference, attenuation, fading
  - ➢ Bit synchronization problems
- More frequent when SNR is low
- More frequent on wireless links
- Measure: *bit error rate*



41

Duke UNIVERSITY

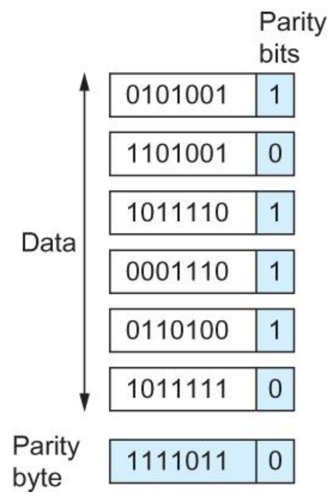
Error Detection

- Error detection code adds redundancy
  - ➢ Analogy: sending two copies
  - ➢ Goal: Maximize the probability of detecting errors *using only a small number of redundant bits*
- Error detection approaches: parity, checksum, CRC

Duke UNIVERSITY

Two-dimensional Parity (1/2)

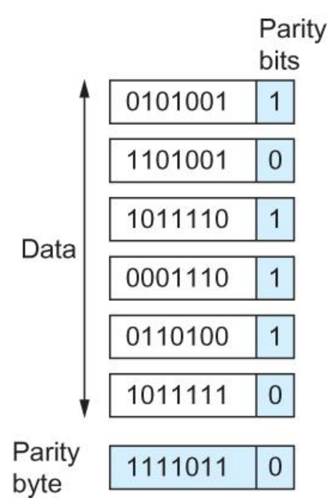
- Even parity bit
  - Make an even number of 1s in each row and column
- 1 extra bit for each byte and one extra byte for the frame
- Detect all 1,2,3-bit errors, and most 4-bit errors



7*6 = 42 bits of payload

Two-dimensional Parity (2/2)

- What is the overhead of this scheme?
  - 14 extra bits
  - 14/42*100 = 33%



7*6 = 42 bits of payload

Internet Checksum Algorithm

- Used by higher-layer protocols
- Basic idea:
  - Add all the words transmitted, send the sum
  - Receiver does the same computation and compares the sums
- IP checksum
  - Adding 16-bit short integers using 1's complement arithmetic
  - Take 1's complement of the result
- Used by lab 2 to detect errors

Duke UNIVERSITY

1's Complement Arithmetic (1/2)

- "-x" is each bit of "x" inverted: +5: 0101, -5: 1010
- $[-(2^{n-1}-1), 2^{n-1}-1]$ – e.g., n = 3 → [-3, 3]
  - What is the range for n=4 bits?
  - $[-(2^3-1), 2^3-1] = [-7, 7]$

Duke UNIVERSITY

1’s Complement Arithmetic (2/2)

- “-x” is each bit of “x” inverted: +5: 0101, -5: 1010
- If there is a carry bit, add 1 to the sum
- Example: 4-bit integer
  - -5 + -2
  - +5: 0101; -5: 1010;
  - +2: 0010; -2: 1101;
  - -5 + -2 = 1010+1101 = 0111 + one carrier bit;
  - →1000 = -7

Duke UNIVERSITY

Verifying the Checksum

- Adds all 16-bit words together, including the checksum
- 0: correct
- 1: errors

Duke UNIVERSITY

Checksum: Remarks

- Only 16 redundant bits for message of any size
- Not particularly robust
  - Can detect 1 bit errors
  - Not all two-bits
    - Why not?
- Efficient for software implementation

Duke UNIVERSITY

Cyclic Redundancy Check

- Cyclic error-correcting codes
- High-level idea:
  - Represent an $n+1$-bit message with an n degree polynomial $M(x)$: (1,0,0,1) → $1x^3+0x^2+0x^1+1$
  - Divide the polynomial by a degree-k divisor polynomial $C(x)$
  - k-bit CRC: remainder
  - Send message + CRC that is divisible by $C(x)$

Duke UNIVERSITY

Polynomial Arithmetic Modulo 2

- $B(x)$ can be divided by $C(x)$ if $B(x)$ has higher degree
- $B(x)$ can be divided once by $C(x)$ if of same degree
  - ➢ $x^3 + 1$ can be divided by $x^3 + x^2 + 1$
  - ➢ The remainder would be $0^* x^3 + 1^* x^2 + 0^* x^1 + 0^* x^0$ (**obtained by XORing the coefficients of each term**)

Duke UNIVERSITY

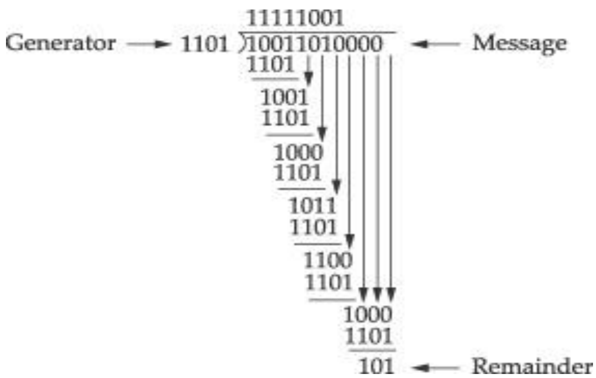
CRC Algorithm

1. Multiply $M(x)$ by x^k: add k zeros the end of the message. Call it $T(x)$
2. Divide $T(x)$ by $C(x)$ and find the remainder
3. Send $P(x) = T(x)$ – remainder
   - ➢ Append remainder to $T(x)$

- $P(x)$ divisible by $C(x)$

Duke UNIVERSITY

An Example



- 8-bit message
  - 10011010
- Divisor (3 bit CRC)
  - 1101

Message sent: **10011010101**

Duke UNIVERSITY

How to Choose a Divisor

- Arithmetic of a finite field
- Intuition: unlikely to be divided evenly by an error
- Corrupted message is $P(x) + E(x)$
- If $E(x)$ is single bit, then $E(x) = x^i$
- If $C(x)$ has the first and last term nonzero, then detects all single bit errors
- Find $C(x)$ by looking it up in a book

Duke UNIVERSITY

International Standards

- CRC-8 = x^8+x^2+x+1
- CRC-10 = $x^{10}+x^9+x^5+x^4+x+1$
- 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 = $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$
  - Used by Ethernet

Duke UNIVERSITY

Lecture Outline

- Link layer: an introduction
- Encoding
- Framing
- Error detection
  - Parity, checksum, CRC
- **Reliable transmission**
  - **FEC, sliding window**

56

Duke UNIVERSITY

Reliable Transmission

- What to do if a receiver detects bit errors?
- Two high-level approaches
  - Forward error correction (FEC)
  - Retransmission

Forward Error Correction

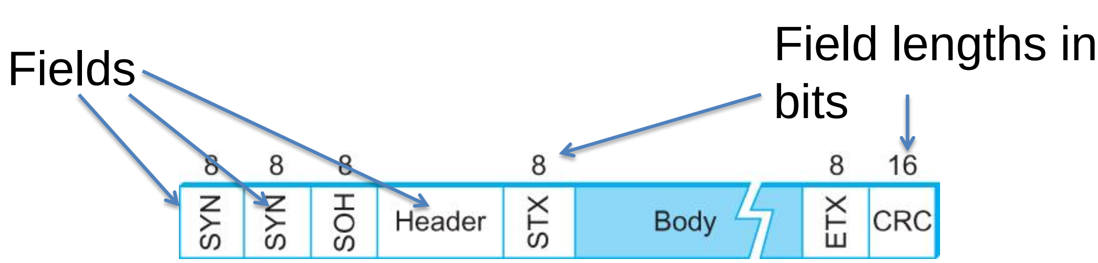
- Uses error-correcting codes
- Incurs fixed overhead for all transmissions
  - As opposed to retransmissions when errors have happened
- Used where retransmissions are costly or impossible
  - One-way communication links
  - Multicast

Reliable Transmission: Retransmissions

- Acknowledgements
  - ACK: A short sequence send back to the sender
  - Can be "piggybacked" on data packets
- Timeouts: waiting for a reasonable time
- Also called *Automatic Repeat reQuest* (ARQ)

Duke UNIVERSITY

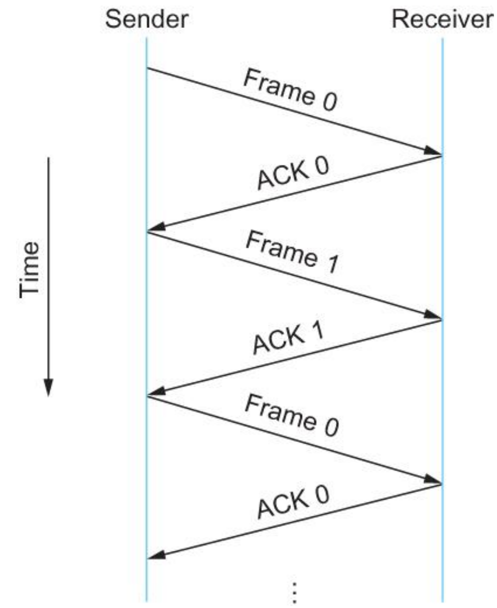
Reminder: Convention: Illustrating Frame and Packet Formats



60

Duke UNIVERSITY

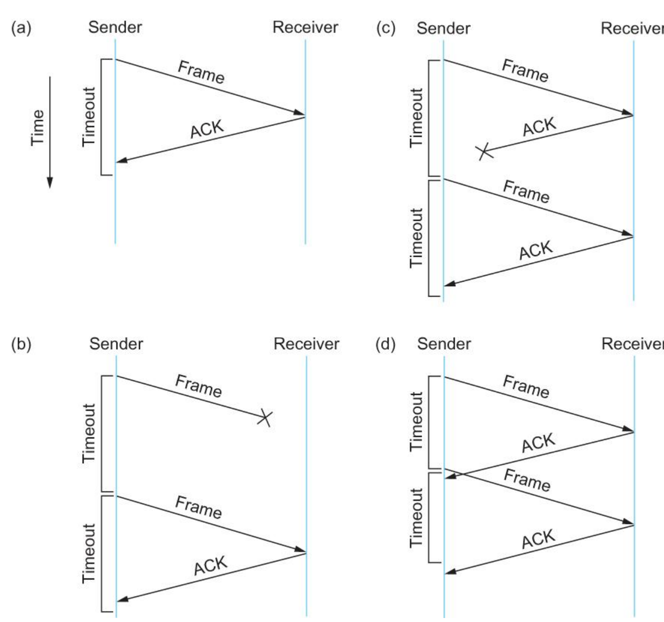
Convention: Illustrating Protocol Behavior



Duke UNIVERSITY

Stop-and-Wait

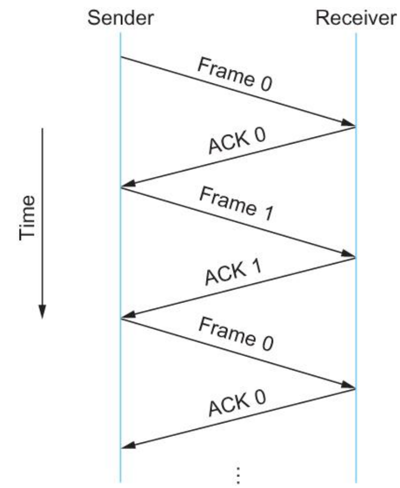
- Send one frame, wait for an ACK, and send the next
- Retransmit if times out
- Note in the last figure (d), there might be confusion: a new frame, or a duplicate?



Duke UNIVERSITY

Sequence Number

- Add a sequence number to each frame to avoid the ambiguity
- Keep sequence numbers small to reduce overhead



Duke UNIVERSITY

Stop and Wait Protocol Drawback

- The sender has only one outstanding frame on the link at a time
  - ➢ This may be **far below** the link's capacity

Duke UNIVERSITY

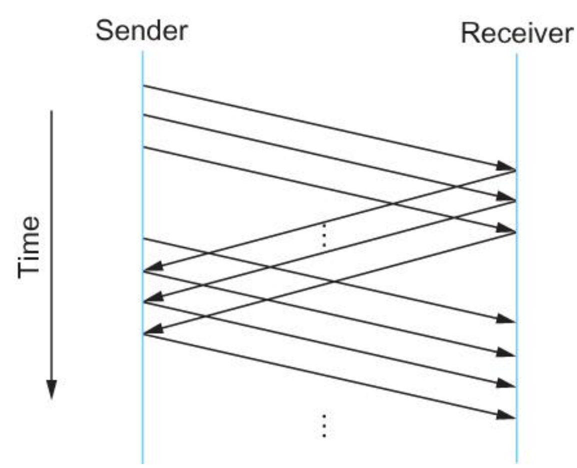
Stop and Wait Protocol Drawback: An Example

- Consider a 1.5 Mbps link with a 45 ms RTT
  - ➢ The link has a delay × bandwidth product of 67.5 Kb or approximately 8 KB
  - ➢ Since the sender can send only one frame per RTT and assuming a frame size of 1 KB
  - ➢ Maximum sending rate
    - Bits per frame ÷ Time per frame = 1024 × 8 ÷ 0.045 = **182 Kbps**
      Or about **one-eighth** of the link's capacity
  - ➢ To use the link fully, then sender should transmit up to eight frames before having to wait for an acknowledgement

Duke UNIVERSITY

Sliding Window

- Key idea: allowing multiple outstanding (un-acked) frames to "*keep the pipe full*"

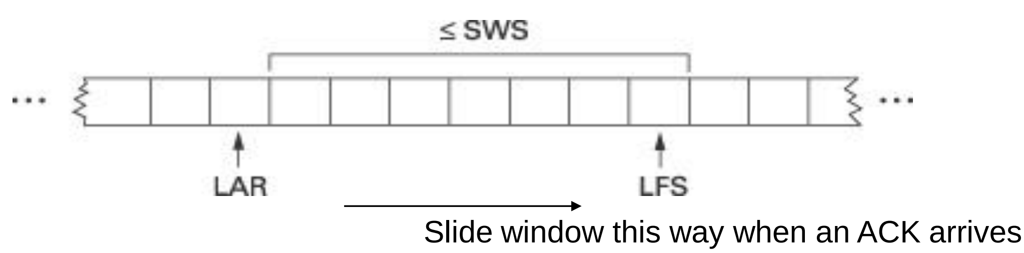


Duke UNIVERSITY

Sliding Window on Sender

- Assign a sequence number (SeqNum) to each frame
- Maintains three variables
  - Send Window Size (SWS)
  - Last ACK Received (LAR)
  - Last Frame Sent (LFS)
- Invariant: LFS – LAR ≤ SWS

Duke UNIVERSITY

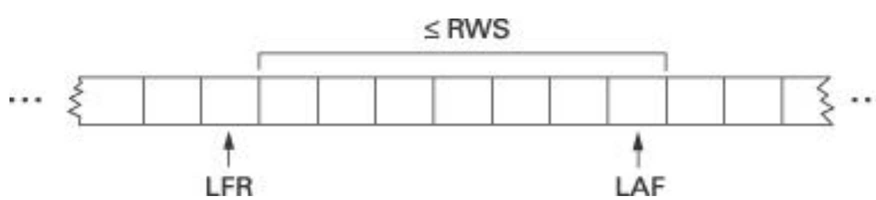


- Sender actions
  - When an ACK arrives, moves LAR to the right, opening the window to allow the sender to send more frames
  - If a frame times out before an ACK arrives, retransmit
    - Frames are buffered

Duke UNIVERSITY

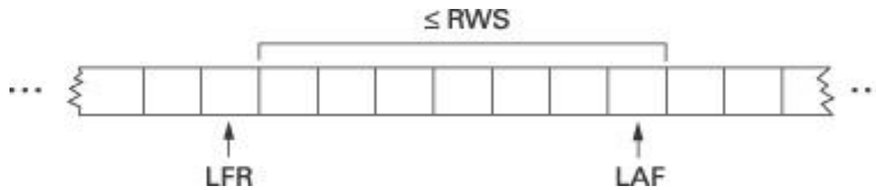
Sliding Window on Receiver

- Maintains three window variables
  - ➢Receive Window Size (RWS)
  - ➢Largest Acceptable Frame (LAF)
  - ➢Last Frame Received (LFR)
- Invariant
  - ➢LAF – LFR ≤ RWS



- A frame with SeqNum arrives...
- Discards it if out of window: SeqNum ≤ LFR or SeqNum > LAF
- If in window, accept, decide on what to ACK
  - ➢ Cumulative ACK
  - ➢ *SeqNumToAck* : largest sequence number not yet acknowledged
  - ➢ Ack *SeqNumToAck*
  - ➢ Sets LFR = *SeqNumToAck*-1, LAF = LFR + RWS
  - ➢ Updates SeqNumToAck

Sliding Window on Receiver: An Example



- Ex: LFR = 5; RWS = 4
  - Implies LAF = 9
- Frames 7, 8, 6 arrive

Finite Sequence Numbers

- Sequence number fields are short
  - They wrap around
- Number of possible sequence numbers must be larger than the maximum number of outstanding frames
  - E.g., stop-and-wait: 1 outstanding frame, 2 sequence numbers

Finite Sequence Numbers: An Example

- Things may go wrong when SWS=RWS, SWS too large
- Example:
  - ➢ 3-bit sequence number [0, … 7], SWS=RWS=7
  - ➢ Sender sends 0, …, 6; receiver ACKs, expects (7,0, …, 5), but all ACKs lost
  - ➢ Sender retransmits 0,…,6; receiver thinks they are new
    - (0, … , 5) are in receiver's window size and will be recorded as parts of a new frame

Duke UNIVERSITY

Finite Sequence Numbers: Window Size Limits

- SWS < (MaxSeqNum+1)/2
  - ➢ Alternates between first half and second half of sequence number space
  - ➢ Just as stop-and-wait alternates between 0 and 1

Duke UNIVERSITY

Multiple Functions of the Sliding Window Algorithm

- Perhaps one of the best-known algorithms in computer networking
- Multiple functions
  - ➢ Reliable delivery of frames over a link
  - ➢ In-order delivery to upper layer protocol
  - ➢ Flow control
    - Not to over-run a slow receiver
  - ➢ Congestion control (later)
    - Not to congest the network

Duke UNIVERSITY

Other ACK Mechanisms

- NACK: negative ACKs for packets not received
  - ➢ Unnecessary, as sender timeouts would catch this information
- SACK: selective ACK the received frames
  - ➢ Refinement to cumulative ACKs: specify exactly what has been received
  - ➢ + No need to send duplicate packets
  - ➢ - more complicated to implement
  - ➢ Newer version of TCP has SACK

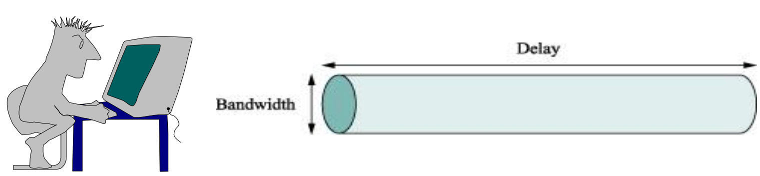
Duke UNIVERSITY

Concurrent Logical Channels

- A link has multiple logical channels
- Each logical channel runs an independent stop-and-wait protocol
- + keeps the pipe full
- - no relationship among the frames sent in different channels: *wildly* out-of-order

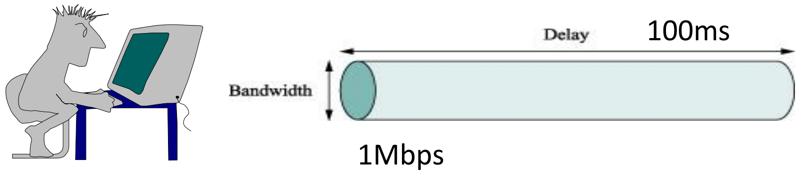
Duke UNIVERSITY

Exercise



- Delay: 100ms; Bandwidth: 1Mbps; Packet Size: 1000 Bytes; ACK: 40 Bytes
- Q: the smallest window size to keep the pipe full?

Duke UNIVERSITY



- Window size = largest amount of unacked data
- How long does it take to ACK a packet?
  - RTT = 100 ms * 2 + transmission delay of a packet (1000 B) + transmission delay of an ACK (40 B) ~=208ms
- How many packets can the sender send in an RTT?
  - 1 Mbps * 208 ms / 8000 bits = 26
- Roughly 13 packets in the pipe from sender to receiver, and 13 ACKs from receiver to sender

Duke UNIVERSITY

Lecture Summary

- Link layer: an introduction
- Encoding
- Framing
- Error detection
  - Parity, checksum, CRC
- Reliability
  - FEC, sliding window

80

Duke UNIVERSITY

Next Lecture

- Wireless communications
- Link layer: Ethernet
- Link layer: Multiple access