15-441 15-641 Computer Networking

Lecture 24 – Datalink

Peter Steenkiste

Fall 2016

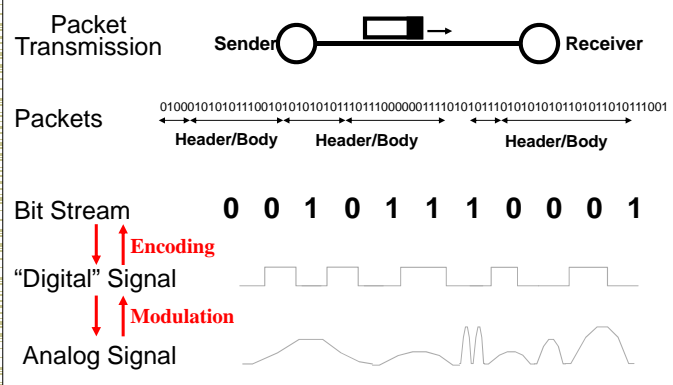www.cs.cmu.edu/~prs/15-441-F16

Outline

- Encoding and decoding
  - Translate between bits and digital signal
- Framing
  - Bit stream to packets
- Packet loss & corruption
  - Error detection
  - Flow control
  - Loss recovery

2

From Signals to Packets

Packet Transmission — Sender → Receiver

Packets: 0100010101011100101010101011011110000001111010101110101010101010101010111001

Header/Body   Header/Body   Header/Body

Bit Stream: 0 0 1 0 1 1 1 0 0 0 1

Encoding ↕

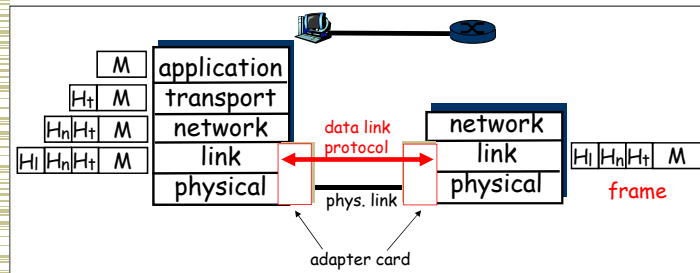"Digital" Signal

Modulation ↕

Analog Signal

3

Link Layer: Implementation

- Implemented in "adapter"
  - E.g., Ethernet card or chip
  - Typically includes: RAM, DSP chips, host bus interface, and link interface

M	application
H_t M	transport
$H_n H_t$ M	network
$H_l H_n H_t$ M	link
	physical

data link protocol

network

link

physical

phys. link

$H_l H_n H_t$ M   frame

adapter card

4

Datalink Functions

- Encoding: change bit stream before transmission
- Framing: encapsulating a network layer datagram into a bit stream.
  - Add header, mark and detect frame boundaries
- Error control: error detection and correction to deal with bit errors.
  - May also include other reliability support, e.g. retransmission
- Flow control: avoid that sender outruns the receiver
- Media access: controlling which frame should be sent next over datalink.
- Hubbing, bridging: extend the size of the network
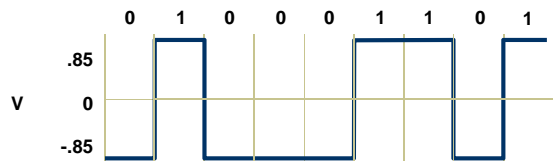
5

Why Do We Need Encoding?

- Keep receiver synchronized with sender.
- Create control symbols, in addition to regular data symbols.
  - E.g. start or end of frame, escape, ...
- Error detection or error corrections.
  - Some codes are illegal so receiver can detect certain classes of errors
  - Minor errors can be corrected by having multiple adjacent signals mapped to the same data symbol
- Encoding can be done one bit at a time or in multi-bit blocks, e.g., 4 or 8 bits.
- Encoding can be very complex, e.g. wireless

6

How to Encode?

- Seems obvious, why waste time on this?   Just modulate the signal!

```
      0    1    0    0    0    1    1    0    1
.85
V   0
-.85
```

- But:
  - How easily can the receiver retrieve the bit stream?
  - What happens when there are errors: a bit gets flipped?
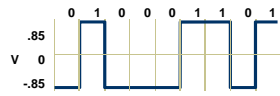
7

How about the Poor Receiver?

```
      0    1    0    1    How many more ones?
```
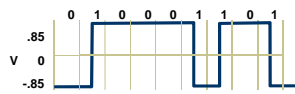
- Sender needs to help the receiver by "shaping" the digital bit stream so it easy to correctly interpret
  - Applies to combination of modulation and coding
- Problem in this case: not enough transitions

8

Simple Encoding Examples

0 1 0 0 0 1 1 0 1

.85
V 0
-.85

Non-Return to Zero (NRZ)

0 1 0 0 0 1 1 0 1

.85
V 0
-.85

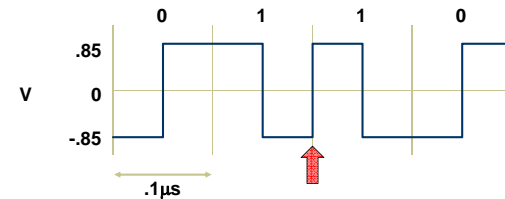Non-Return to Zero Inverted (NRZI)

0 1 1 0

.85
V 0
-.85

Manchester

- Change the bit stream so there are enough transitions in the signal
  - Helps the receiver
- But this also increases the complexity of the signal
  - Uses more spectrum!
- Manchester encoding was used in the original ethernet

Manchester Encoding

0   1   1   0

.85
V 0
-.85

.1µs

- Used by Ethernet
- 0=low to high transition, 1=high to low transition.
- Transitions simplify clock recovery and good electrical properties for any bit stream
- But you pay a price!
  - Doubles the number of transitions – more spectrum!
  - Circuitry must run twice as fast

Take-away: Encoding and Modulation

- Encoding and modulation work together
  - Must generate a signal that works well for the receiver – has good electrical properties
  - Must be efficient with respect to spectrum use
  - Can shift some of the burden between the two layers
  - Tradeoff is figured out by our electrical engineers
- Maintaining good electrical properties
  - Spectrum efficient modulation requires more encoding
  - For example: 4B/5B encoding (later)
- Error recovery
  - Aggressive modulation needs stronger coding

Outline

- Encoding and decoding
  - Translate between bits and digital signal
- Framing
  - Bit stream to packets
- Packet loss & corruption
  - Error detection
  - Flow control
  - Loss recovery

Framing

Access ctrl	Frame ctrl	Dest adr	Src adr	Body	checksum

0100010101011001010101010111011100000011110101011101010101011010110101111001

Start delim	Access ctrl	Frame ctrl	Dest adr	Src adr	Body	checksum	End delim

---

Framing

- A link layer function, defining which bits have which function.
- Minimal functionality: mark the beginning and end of packets (or frames).
- Some techniques:
  - Out of band delimiters (e.g. 4B/5B control symbols)
  - Frame delimiter characters with character stuffing
  - Frame delimiter codes with bit stuffing
  - Synchronous transmission (e.g. SONET)
    - Boundaries are based on timing

Delimiter Based

- SYN: sync character
- SOH: start of header
- STX: start of text
- ETX: end of text

- What happens when ETX is in Body?

SYN	SYN	SOH	Header	STX	Body	ETX	CRC

Character and Bit Stuffing

- Mark frames with special character.
  - What happens when the user sends this character?
  - Use escape character when a control symbol appears in data:
  - *abc*def →*abc*def
  - Very common on serial lines, in editors, etc.
- Mark frames with special bit sequence (read)
  - must ensure data containing this sequence can be transmitted
  - example: suppose 11111111 is a special sequence.
  - transmitter inserts a 0 when this appears in the data:
  - 11111111 → 111111101 – receiver deletes a 0 after seven 1's
  - Means that we must stuff a zero any time seven 1s appear:
  - 11111110 → 111111100
  - receiver unstuffs: 111111100 → 11111110

Ethernet Framing

- Preamble is 7 bytes of 10101010 (5 MHz square wave) followed by one byte of 10101011
- Allows receivers to recognize start of transmission after idle channel

preamble	datagram	length	more stuff

17

4B/5B Encoding

- Data coded as symbols of 5 line bits → 4 data bits, so 100 Mbps uses 125 MHz.
- Encoding ensures no more than 3 consecutive 0's
- Uses NRZI to encode resulting sequence
  - Uses less frequency than Manchester encoding
- 16 data symbols, 8 control symbols
  - Data symbols: 4 data bits
  - Control symbols: idle, begin frame, etc.
- Example: FDDI.

18

4B/5B Encoding

Data	Code	Data	Code
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

From datalink

To modulator

19

Other Encodings

- 8B/10B: Fiber Channel and Gigabit Ethernet
- 64B/66B: 10 Gbit Ethernet (& 40 and 100 Gb/S)
- B8ZS: T1 signaling (old - bit stuffing)

Things to Remember

- Encoding and modulation must work together
- Lots of approaches
- Rule of thumb:
  - Little bandwidth → complex encoding
  - Lots of bandwidth → simple encoding

20

5

Outline

- Encoding
  - Bits to digital signal
- Framing
  - Bit stream to packets
- Packet loss & corruption
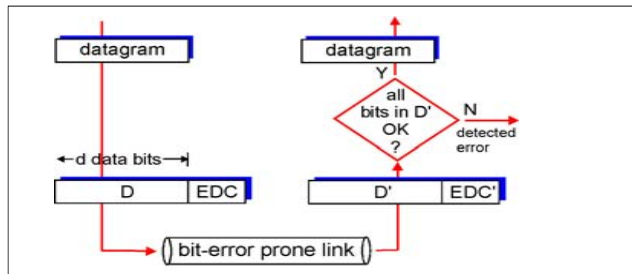 - Error detection
  - Flow control
  - Loss recovery

Error Coding

- Transmission may introduce errors into a message.
  - Received "digital signal" is different from that transmitted
  - Single bit errors versus burst errors
- Detection:
  - Requires a convention that some messages are invalid
  - Hence requires extra bits
  - An (n,k) code has codewords of n bits with k data bits and r = (n-k) redundant check bits
- Correction
  - Forward error correction: many related code words map to the same data word
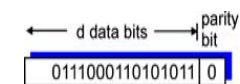 - Detect errors and retry transmission

Error Detection

- EDC= Error Detection and Correction bits (redundancy)
- D   = Data protected by error checking, may include header fields
- Error detection not 100% reliable!
  - Protocol may miss some errors, but rarely
  - Larger EDC field yields better detection and correction

Parity Checking

Single Bit Parity:
Detect single bit errors

Internet Checksum

- Goal: detect "errors" (e.g., flipped bits) in transmitted segment

Sender	Receiver

- Treat segment contents as sequence of 16-bit integers
- Checksum: addition (1's complement sum) of segment contents
- Sender puts checksum value into checksum field in header
	- Compute checksum of received segment
	- Check if computed checksum equals checksum field value:
	  - NO - error detected
	  - YES - no error detected. But maybe errors nonethless?

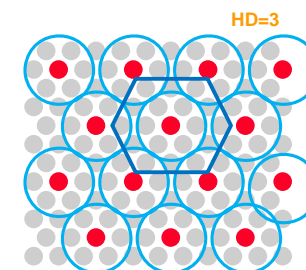---

Basic Concept: Hamming Distance

- Hamming distance of two bit strings = number of bit positions in which they differ.

  1 0 1 1 0
  1 1 0 1 0   **HD=2**

- If the valid words of a code have minimum Hamming distance D, then D-1 bit errors can be detected.

  HD=3

- If the valid words of a code have minimum Hamming distance D, then $[(D-1)/2]$ bit errors can be corrected.

Cyclic Redundancy Codes (CRC)

- Commonly used codes that have good error detection properties.
  - Can catch many error combinations with a small number of redundant bits
- Based on division of polynomials.
  - Errors can be viewed as adding terms to the polynomial
  - Should be unlikely that the division will still work
- Can be implemented very efficiently in hardware.
- Examples:
  - CRC-32: Ethernet
  - CRC-8, CRC-10, CRC-32: ATM

CRC: Basic idea

- Treat bit strings as polynomials:

  1 0 1 1 1
  X^{4+} $X^2+X^1+X^0$

- Sender and Receiver agree on a *divisor* polynomial of degree k
- Message of M bits → send M+k bits
- No errors if M+k is divisible by divisor polynomial
- If you pick the right divisor you can:
  - Detect all 1 & 2-bit errors
  - Any odd number of errors
  - All Burst errors of less than k bits
  - Some burst errors >= k bits

Outline

- Encoding
  - Bits to digital signal
- Framing
  - Bit stream to packets
- Packet loss & corruption
  - Error detection
  - Flow control
  - Loss recovery

What is Used in Practice?

- No flow or error control.
  - E.g. regular Ethernet, just uses CRC for error detection
- Flow control only
  - E.g. Gigabit Ethernet
- Flow and error control.
  - E.g. X.25 (older connection-based service at 64 Kbs that guarantees reliable in order delivery of data)

- Flow and error control solutions also used in higher layer protocols
  - E.g., TCP for end-to-end flow and error control