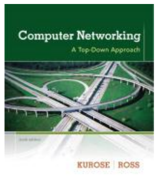
Chapter 5
Link Layer

A note on the use of these ppt slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a lot of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR



Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

The course notes are adapted for Bucknell's CSCI 363
Xiannong Meng
Spring 2016

Chapter 5: Link layer

our goals:

- understand principles behind link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - local area networks: Ethernet, VLANs
- instantiation, implementation of various link layer technologies

Link layer, LANs: outline

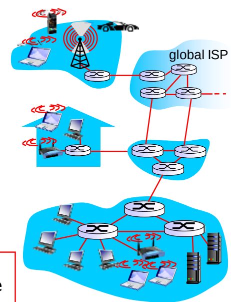
5.1 introduction, services
5.2 error detection, correction
5.3 multiple access protocols
5.4 LANs
- addressing, ARP
- Ethernet
- switches
- VLANS

5.5 link virtualization: MPLS
5.6 data center networking
5.7 a day in the life of a web request

Link layer: introduction

terminology:

- hosts and routers: nodes
- communication channels that connect adjacent nodes along communication path: links
  - wired links
  - wireless links
  - LANs and WANs
- layer-2 packet: *frame,* encapsulates datagram



data-link layer has responsibility of transferring datagram from one node to *physically adjacent* node over a link

Internet protocol stack

- *application:* supporting network applications
  - FTP, SMTP, HTTP
- *transport:* process-process data transfer
  - TCP, UDP
- *network:* routing of datagrams from source to destination
  - IP, routing protocols
- *link:* data transfer between neighboring network elements
  - Ethernet, 802.11 (WiFi), PPP
- *physical:* bits "on the wire"

application
transport
network
link
physical

Link layer: context

- datagram transferred by different link protocols over different links:
  - e.g., Ethernet (802.3) on first link, frame relay on intermediate links, 802.11 on last link
- each link protocol provides different services
  - e.g., may or may not provide *rdt* over link

transportation analogy:

- trip from Princeton to Lausanne
  - limo: Princeton to JFK
  - plane: JFK to Geneva
  - train: Geneva to Lausanne
- tourist = datagram
- transport vehicle = communication link
- transportation procedure = link layer protocol
- travel agent = routing algorithm

Link layer services

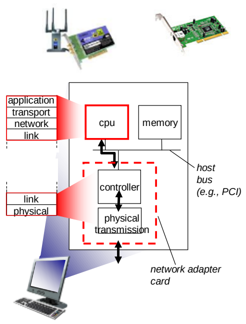
- *framing, link access:*
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - "MAC" addresses used in frame headers to identify source, dest
    - different from IP address!
- *reliable delivery between adjacent nodes*
  - we learned how to do this already (chapter 3)!
  - seldom used on low bit-error link (fiber, some twisted pair)
  - wireless links: high error rates
    - *Q:* why both link-level and end-to-end reliability?

Link layer services (more)

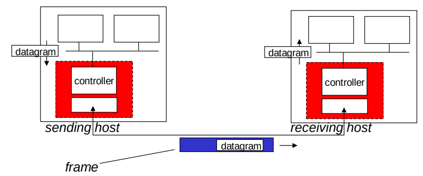
- *flow control:*
  - pacing between adjacent sending and receiving nodes
- *error detection:*
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame
- *error correction:*
  - receiver identifies *and corrects* bit error(s) without resorting to retransmission
- *half-duplex and full-duplex*
  - with half duplex, nodes at both ends of link can transmit, but not at same time

Where is the link layer implemented?

- in each and every host
- link layer implemented in "adaptor" (a.k.a. *network interface card* NIC) or on a chip
  - Ethernet card, 802.11 card; Ethernet chipset
  - implements link, physical layer
- attaches into host's system buses
- combination of hardware, software, firmware



Adaptors communicating



- sending side:
  - encapsulates datagram in frame
  - adds error checking bits, rdt, flow control, etc.
- receiving side
  - looks for errors, rdt, flow control, etc
  - extracts datagram, passes to upper layer at receiving side

Link layer, LANs: outline

5.1 introduction, services

5.2 error detection, correction

5.3 multiple access protocols

5.4 LANs
- addressing, ARP
- Ethernet
- switches
- VLANS

5.5 link virtualization: MPLS

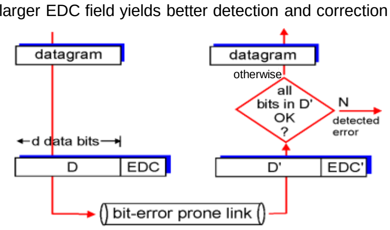
5.6 data center networking

5.7 a day in the life of a web request

Error detection

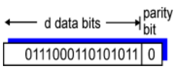
EDC= Error Detection and Correction bits (redundancy)
D = Data protected by error checking, may include header fields

- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction



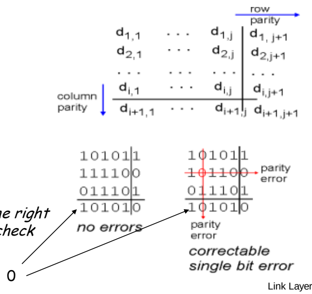
Parity checking

single bit parity:
- detect single bit errors



The above is an odd-parity

two-dimensional bit parity:
- detect and correct single bit errors



What can 2-D parity check do? (1)

- Correct all 1-bit errors (we already saw it);
- Detect all 2-bit errors;
  - An even parity example with two bit errors

```
data              0110100 1     row parities
                  1011010 0
                  0000111 0
                  1110101 1
                  1001011 0

column parities   1000110 1
```

Odd number of 1's in 3rd and 7th columns indicate errors! We know the errors are in these two columns, not sure which row.

http://www.ccs.neu.edu/home/amislove/teaching/cs4700/spring11/lectures/lecture13.pdf

What can 2-D parity check do? (2)

- Detect all 3-bit errors;
  - An even parity example with three bit errors

```
data              0110100 1     row parities
                  1011010 0
                  0000111 0
                  1100101 1
                  1001011 0

column parities   1000110 1
```

Odd number of 1's in 7th column indicates errors! We can't tell where the errors are.

What can 2-D parity check do? (3)

- Detect most 4-bit errors;
  - An even parity example with four bit errors that won't be detected

```
data              0110100 1     row parities
                  1011010 0
                  0000111 0
                  1100100 1
                  1001011 0

column parities   1000110 1
```

All row and column parity checks remain valid.

Internet checksum (review)

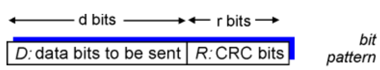
goal: detect "errors" (e.g., flipped bits) in transmitted packet (note: used at transport layer *only*)

sender:
- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

receiver:
- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected. *But maybe errors nonetheless?*

Cyclic redundancy check

- more powerful error-detection coding
- view data bits, D, as a binary number
- choose r+1 bit pattern (generator), G
- goal: choose r CRC bits, R, such that
  - <D,R> exactly divisible by G (modulo 2)
  - receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
  - can detect all burst errors less than r+1 bits
- widely used in practice (Ethernet, 802.11 WiFi, ATM)



$D * 2^r$ XOR R   *mathematical formula*

CRC basics

want: data | CRC | G

$D \cdot 2^r$ XOR R = nG

equivalently:
$D \cdot 2^r$ = nG XOR R

Are we convinced that the above are equivalent?

Because A XOR A = 0, A XOR 0 = A,
and (A XOR B) XOR B = A XOR (B XOR B),
we have ($D \cdot 2^r$ XOR R) XOR R = nG XOR R

CRC example

want:
$D \cdot 2^r$ XOR R = nG

equivalently:
$D \cdot 2^r$ = nG XOR R

equivalently:
if we divide $D \cdot 2^r$ by G, want remainder R to satisfy: (remember 1 XOR A = A)

$$\boxed{R = \text{remainder of } \left[\frac{D \cdot 2^r}{G}\right]}$$

G, D, $D*2^r$, $r = 3$, R

```
        101011
1001)101110000
     1001
      101
      000
      1010
      1001
       110
       000
       1100
       1001
        1010
        1001
        0011
```


Good generating polynomials (IEEE Standards)

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

or

G_{CRC-32} = 1 0000 0100 1100 0001 0001 1101 1011 0111

$G_{CRC-16} = x^{16}+x^{12}+x^{5}+1$

or

G_{CRC-16} = 1 0001 0000 0010 0001

How many bit errors can be detected with CRC? (1)

- Consider message received as T(x)+E(x), where T(x) is the original, correct message, E(x) is the error. Take [T(x)+E(x)]/G(x), because T(x)/G(x) is zero, we only need to focus on E(x)/G(x)
  - If a single bit error, E(x) = x^i, if G(x) has two or more terms, G(x) will never divide E(x), all single bit errors can be detected;
  - If two isolated single bit errors, E(x) = $x^i + x^j$, i > j, we can rewrite E(x) = $x^j(x^{i-j} + 1)$. Assume G(x) is not divisible by x, we can choose G(x) to contain a term not divisible by $(x^{i-j} + 1)$, then all double errors can be detected (e.g., choose *k* to be the length of the frame);

How many bit errors can be detected with CRC? (2)

- If there are odd number of bits in error, E(x) contains an odd number of terms, (e.g., $x^5 + x^2 + 1$, but not x^3+1). No polynomial with an odd number of terms has x+1 as a factor in the modulo 2 system. By making x+1 a factor of G(x), we can detect all errors with odd number of error bits!
- Polynomial code with *r* check bits will detect all burst errors of length <= *r*! A burst error of length k can be written as $x^i(x^{k-i} + \ldots + 1)$. If G(x) contains an x^0 term, it will not have x^i as a factor, thus will not divide E(x) evenly. For example a 16-bit polynomial can detect all burst errors of 16 bits or less.

Computer Networks by A.S. Tanenbaum, 4th edition, Prentice Hall 2003

Why various different types of error checking and correction?

- CRC is effective in detecting and correcting errors and can be easily implemented in hardware (shift registers).
- Internet checksum implementation is in software, mostly at the network and transport layer.