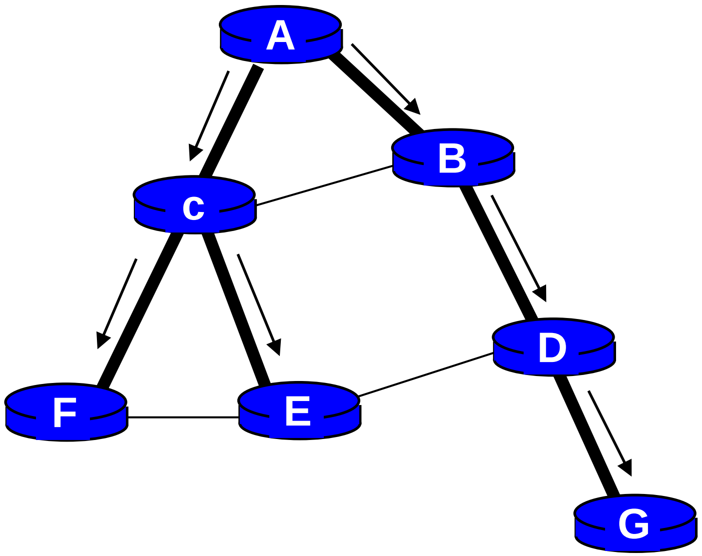
Broadcast, link layer, error detection




```
1 1 0 1 | 1 0 0 1 1 0 1 0 0 0 0
          1 1 0 1
            1 0 0 1
            1 1 0 1
              1 0 0 0
              1 1 0 1
                1 0 1 1
                1 1 0 1
                  1 1 0 0
                  1 1 0 1
                      1 0 0 0
                      1 1 0 1
                        1 0 1
```

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

Chapter 4: outline

4.1 Introduction

4.2 Virtual circuit and datagram networks

4.3 What's inside a router

4.4 IP: Internet Protocol

- Datagram format
- IPv4 addressing
- Network Address Translation (NAT)
- DHCP
- ICMP
- IPv6
- IPsec

4.5 Routing algorithms

- Link state
- Distance vector
- Hierarchical routing

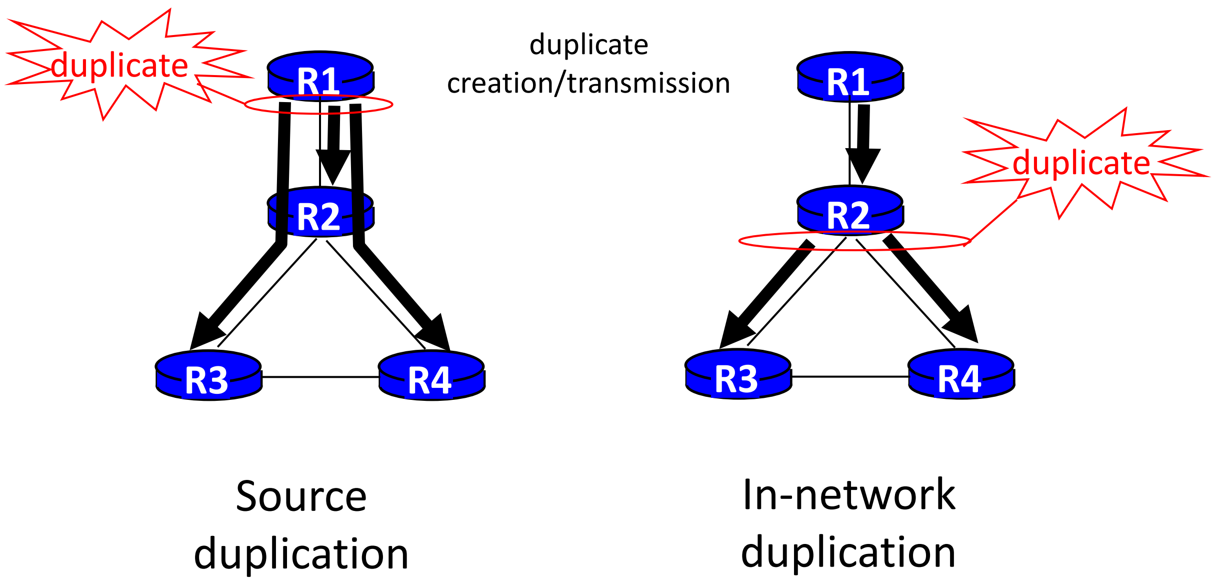
4.6 Routing in the Internet

- RIP
- OSPF
- BGP

4.7 Broadcast and multicast routing

Broadcast routing

- Deliver packets from source to all other nodes
- Source duplication is inefficient:



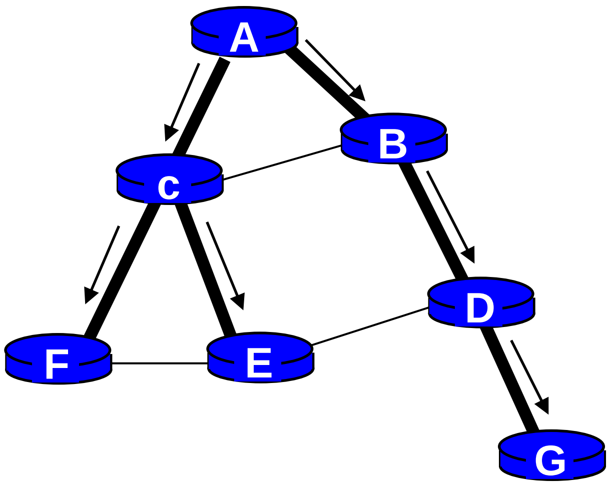
- Source duplication: How does source determine recipient addresses?

In-network duplication

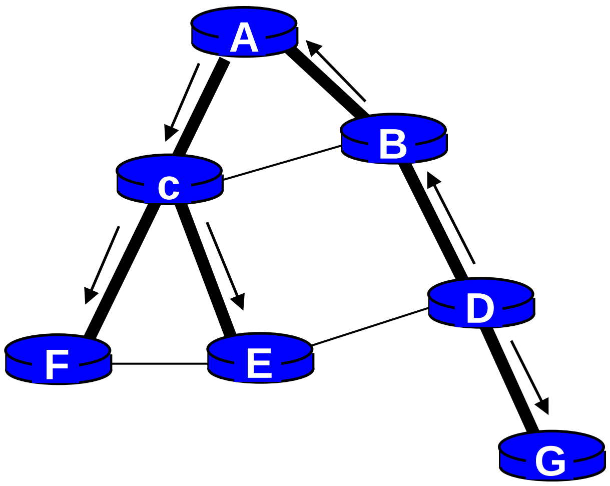
- *Flooding:* When node receives broadcast packet, sends copy to all neighbors
  - Problems: cycles & broadcast storm
- *Controlled flooding:* Only broadcast packet if it hasn't been broadcast before
  - Option 1: Keep track of packet ids already broadcast
  - Option 2: Reverse Path Forwarding (RPF), only forward packet if it arrived on shortest path between node and source
- *Spanning tree:*
  - No redundant packets received by any node

Spanning tree

- First construct a spanning tree
- Nodes forward copies only along spanning tree



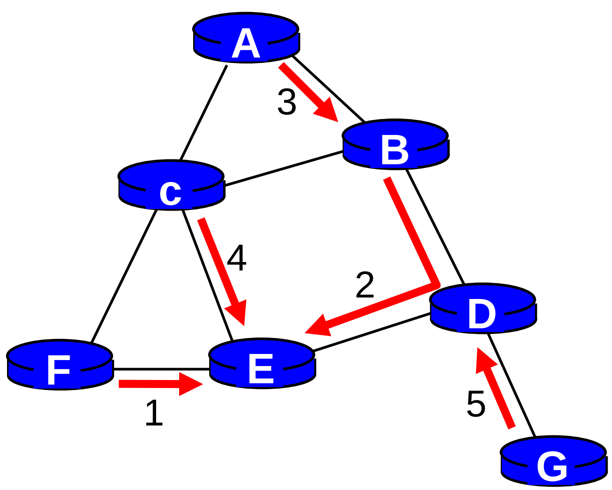
(a) Broadcast initiated at A



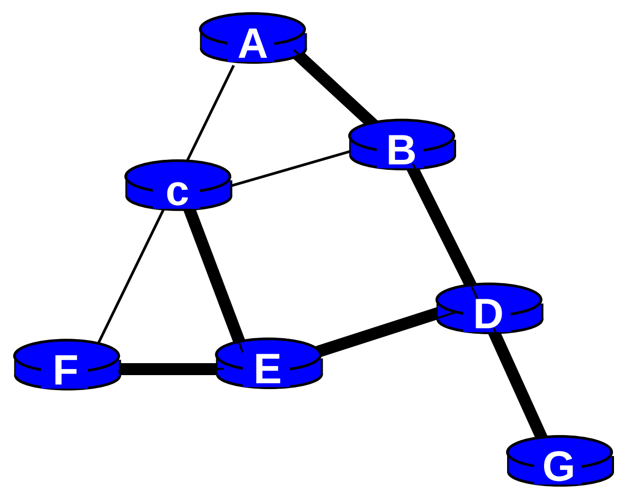
(b) Broadcast initiated at D

Spanning tree: creation

- Center node
- Each node sends unicast join message to center node
  - Message forwarded until it arrives at a node already belonging to spanning tree



(a) Stepwise construction of spanning tree (center: E)



(b) Constructed spanning tree

Spanning tree algorithm

- Problem: loops in the network topology
  - Radia Perlamn at DEC
  - One week to figure out how to join LANs without loops
  - Took one day, then wrote a poem:

I think that I shall never see
A graph more lovely than a tree.
A tree whose crucial property
Is loop-free connectivity.
A tree which must be sure to span.
So packets can reach even LAN.
First the Root must be selected
by ID it is elected.
Least cost paths from Root are traced
In the tree these paths are placed.
A mesh is made by folks like me
Then bridges find a spanning tree.

http://www.youtube.com/watch?v=iE_AbM8ZykI

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

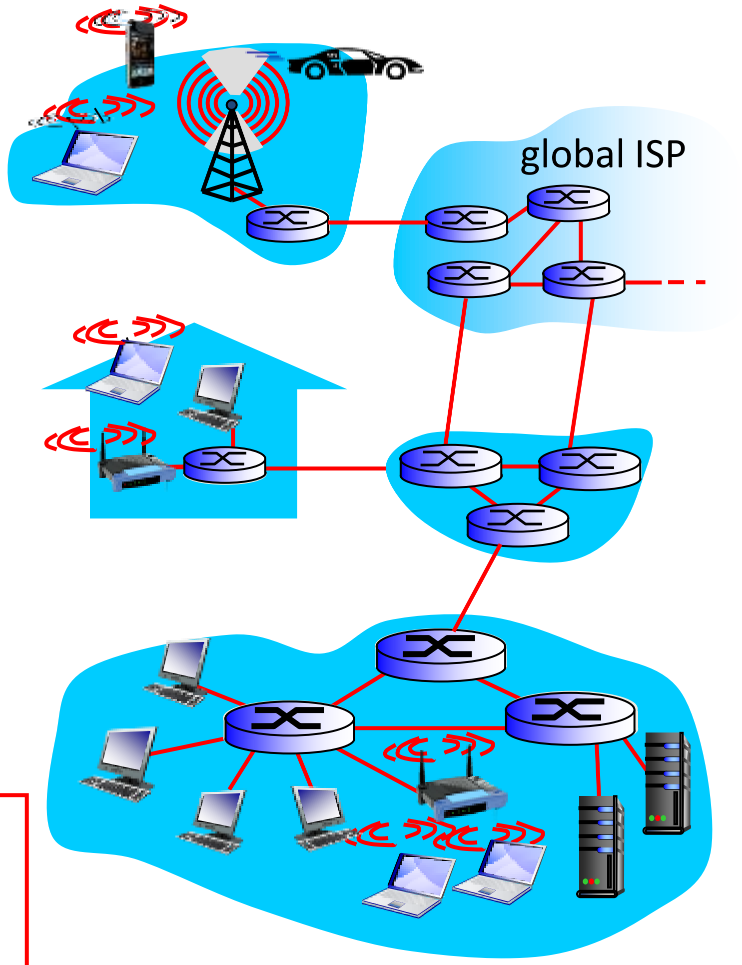
5.6 Data center networking

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
- Layer-2 packet: frame, encapsulates datagram



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

Link layer services

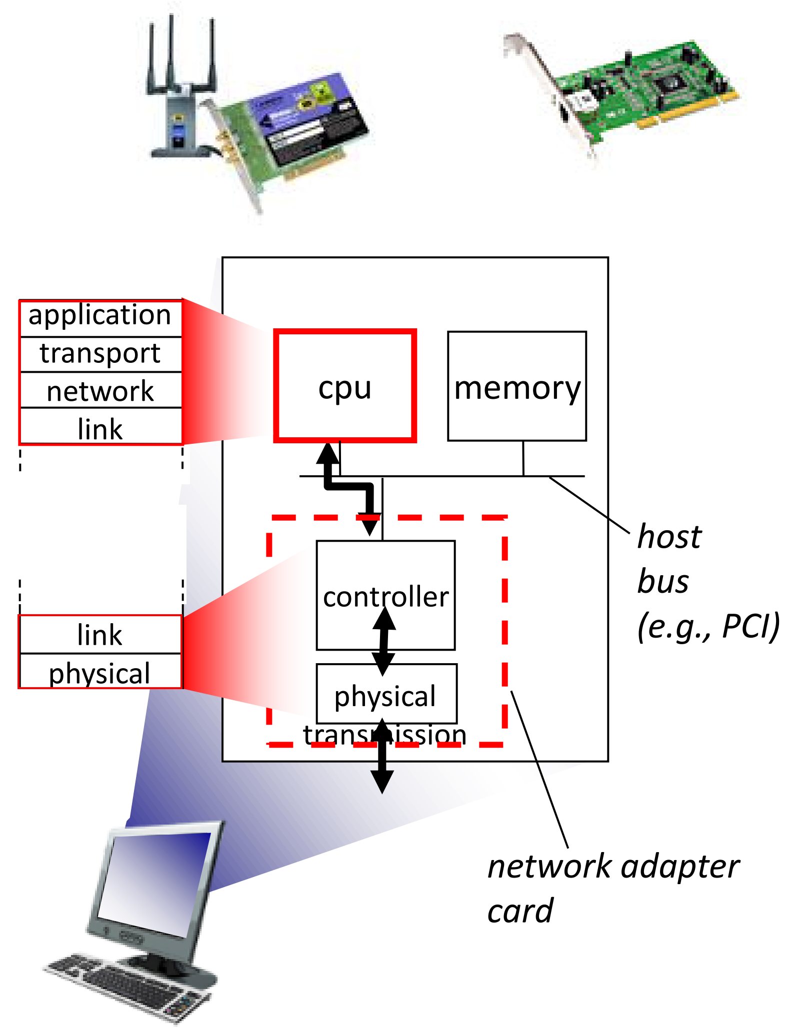
- *Framing, link access:*
  - Encapsulate datagram into frame, adding header
  - Channel access if shared medium
  - MAC addresses used in frame headers to identify source & destination
    - Different from IP address!
- *Reliable delivery between adjacent nodes*
  - We learned how to do this already!
  - Seldom used on low bit-error links (e.g. fiber)
  - Wireless links: high error rates
    - *Q:* Why both link-level and end-end reliability?

Link layer services (more)

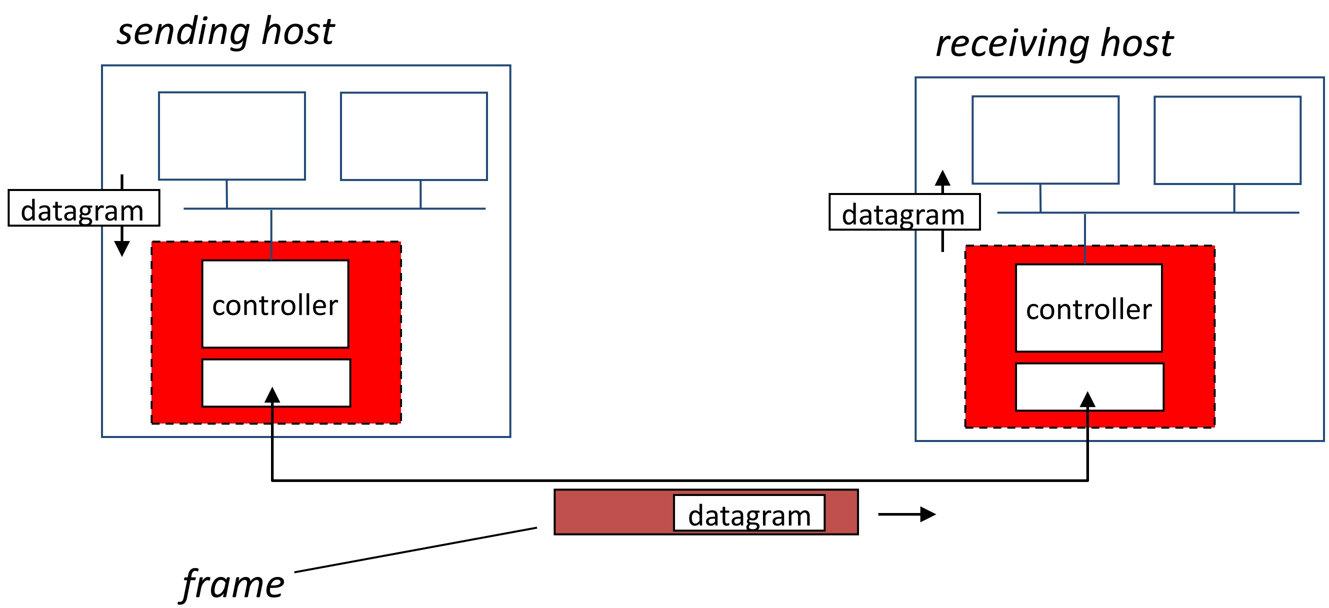
- ***Flow control:***
  - Pacing between adjacent sending and receiving nodes
- ***Error detection*:**
  - Errors caused by signal attenuation, noise.
  - Receiver detects presence of errors:
    - Signals sender for retransmission or drops frame
- ***Error correction:***
  - Receiver identifies *and 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 *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

Error detection

- Error detection
  - Parity checking
  - Checksum
  - Cyclic Redundancy Check
- Error correction
  - Retransmission
  - Forward error correction (ECC)
    - Hamming codes, Reed-Solomon codes, low-density parity check code (LDPC)
    - Examples: DVDs, WiMax, 802.11n

Error detection

- Basic idea: add redundant data
  - Simple scheme:
    - Send two copies of data
    - Compare copies, any differences implies error
    - High overhead, 2n bits to send n bits data
  - More complex schemes:
    - Strong error detection with k redundant bits
    - k << n
    - e.g. Ethernet frame with 12K bits, 32-bit CRC

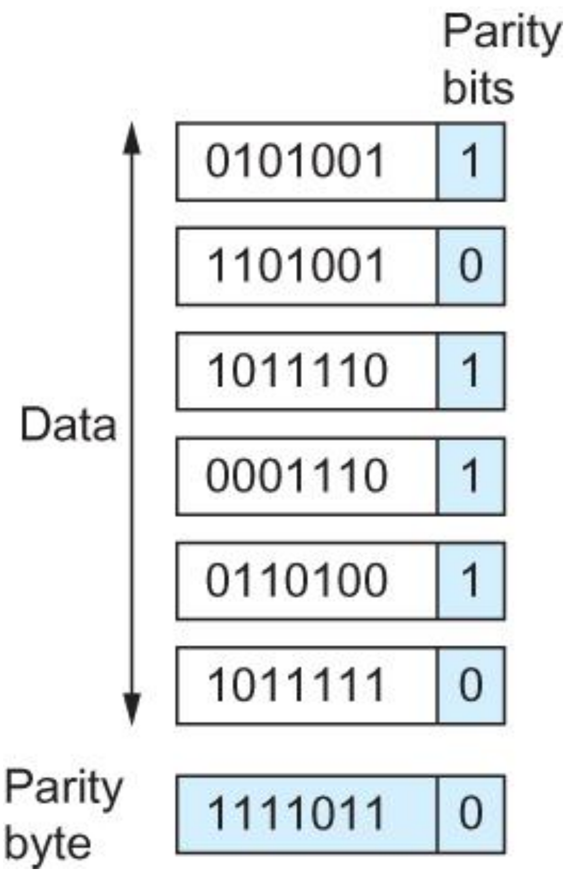
Parity checking

- One dimensional parity
  - Set parity bit so number of 1s odd or even
  - Detects all single bit errors
  - Example (7 bits data, 1 bit data):

data	even parity	odd parity
0010 101	0010 1011	0010 1010
1100 110	1100 1100	1100 1101
0000 000	0000 0000	0000 0001

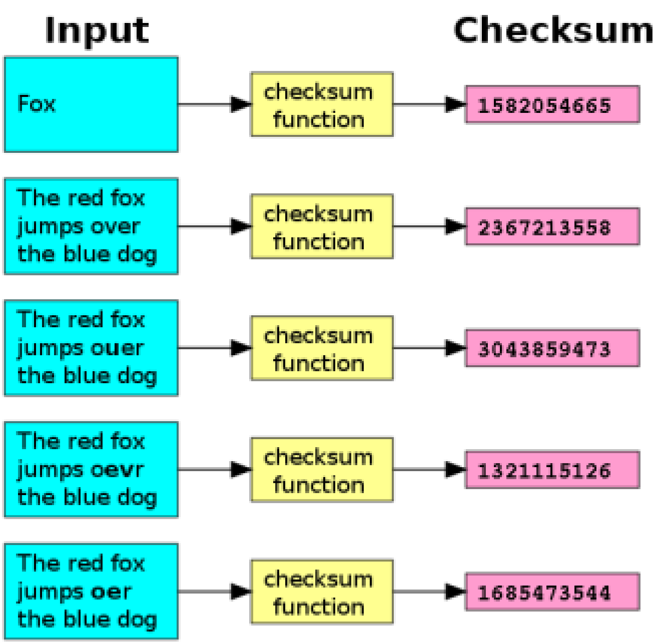
Parity checking

- Two-dimensional parity
  - Arrange bytes in a table
  - Parity over rows and over columns
  - Catches all 1-3 bit errors
  - Catches most 4 bit errors



Checksum

- Internet checksum algorithm
  - Add up 16-bit words and transmit result
  - Not used in link-layer
    - Used in higher layers like TCP and UDP
  - Advantages:
    - Small # of redundant bits
    - Easy to implement
  - Disadvantages:
    - Weak protection



Checksum

- Algorithm:
  - Add data using one's complement
  - Checksum is complement of summation
  - One's complement:
    - Negative #'s are bit complement of positive
    - Carry out from most sig. bit, increment by 1

1	1	1	1	1	1	1		
	0	0	0	0	1	1	1	1
+	1	1	1	1	1	0	1	0
	0	0	0	0	1	0	0	1
+								1
	0	0	0	0	1	0	1	0

	15
+	-5
	10

Checksum algorithm

```
u_short cksum(u_short *buf, int count)
{
        register u_long sum = 0;
        while (count--)
        {
                sum += *buf++;
                if (sum & 0xFFFF0000)
                {
                        /* carry occurred so wrap around */
                        sum &= 0xFFFF;
                        sum++;
                }
        }
        return ~(sum & 0xFFFF);
}
```

Cyclic redundancy check

- CRC (cyclic redundancy check)
  - Maximize detection while minimizing redundancy
  - (n+1) bit message = n degree polynomial

Message (x)	polynomial M(x)
1001 1010	$1x^7 + 0x^6 + 0x^5 + 1x^4 + 1x^3 + 0x^2 + 1x^1 + 0x^0$ $= x^7 + x^4 + x^3 + x^1$

  - Sender/receiver agree on divisor polynomial C(x)
    - C(x) is polynomial of degree k (k << n)
  - Extend message M(x) to include extra bits that make it evenly divisible by C(x)

Common CRC polynomials

Name	Used in	C(x)	Generator
CRC-8	ATM	$x^8 + x^2 + x^1 + 1$	1 0000 0111
CRC-10	ATM	$x^{10} + x^9 + x^5 + x^4 + x^1 + 1$	110 0011 0011
CRC-12	Telecom systems	$x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$	1 1000 0000 1111
CRC-16	USB, Bisync	$x^{16} + x^{15} + x^2 + 1$	1 1000 0000 0000 0011
CRC-CCITT	Bluetooth, X.25, SD, HDLC	$x^{16} + x^{12} + x^5 + 1$	1 0001 0000 0010 0001
CRC-32	Ethernet, SATA, MPEG-2, Gzip, PKZIP, PNG, ATM	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$	1 0000 0100 1100 0001 0001 1101 1011 0111

- CRC will detect:
  - All single-bit errors, if x^k and x^0 are nonzero
  - All double-bit errors, if C(x) has a factor with 3+ terms
  - Any odd number of errors, if C(x) contains the factor (x+1)
  - Any burst error, if burst is less than k bits

Generating a CRC

- Adding a CRC to a message:
  - Assume:
    - Message M(x) of (n+1) bits
    - Generator polynomial C(x) of degree k
  - Add k zeros to right side of M(x)
  - Find remainder by dividing by C(x)
  - Replace k zeros on right of M(X) with remainder

Example CRC generation

Message	1001 1010	$M(x) = x^7 + x^4 + x^3 + x^1$
Generator	1101	$C(x) = x^3 + x^2 + 1$

```
1 1 0 1 | 1 0 0 1 1 0 1 0 0 0 0
          1 1 0 1
          -------
            1 0 0 1
            1 1 0 1
            -------
              1 0 0 0
              1 1 0 1
              -------
                1 0 1 1
                1 1 0 1
                -------
                  1 1 0 0
                  1 1 0 1
                  -------
                        1 0 0 0
                        1 1 0 1
                        -------
                          1 0 1
```

- Repeatedly XOR generator with bits from augmented M(x)
- Final remainder is CRC
- Final message:
  1001 1010 101

Checking a CRC

- Checking a received message:
  - Receiver gets M(x) + calculated CRC
  - Divides by agreed C(x)
  - If remainder = 0, no error detected

Example uncorrupted message

Message	1001 1010	$M(x) = x^7 + x^4 + x^3 + x^1$
Generator	1101	$C(x) = x^3 + x^2 + 1$
CRC	101	

```
1 1 0 1 | 1 0 0 1 1 0 1 0 1 0 1
          1 1 0 1
          -------
            1 0 0 1
            1 1 0 1
            -------
              1 0 0 0
              1 1 0 1
              -------
                1 0 1 1
                1 1 0 1
                -------
                  1 1 0 0
                  1 1 0 1
                  -------
                      1 1 0 1
                      1 1 0 1
                      -------
                            0
```

- Repeatedly XOR generator with bits from message + CRC
- Final remainder should be zero

Example corrupted message

Message	1001 1010	$M(x) = x^7 + x^4 + x^3 + x^1$
Generator	1101	$C(x) = x^3 + x^2 + 1$
CRC	101	

```
        _____________________
1 1 0 1 | 1 0 0 1 1 0 0 0 1 0 1
          1 1 0 1
          -------
            1 0 0 1
            1 1 0 1
            -------
              1 0 0 0
              1 1 0 1
              -------
                1 0 1 0
                1 1 0 1
                -------
                  1 1 1 0
                  1 1 0 1
                  -------
                      1 1 1 0
                      1 1 0 1
                      -------
                          1 1 1
```

- Repeatedly XOR generator with bits from message + CRC
- Final remainder should be zero

Error detection rate

Type	Length	Error detection
checksum	8-bit	99.6094%
checksum	16-bit	99.9985%
CRC	32-bit	99.9999%

Summary

- Broadcast routing
  - Sending message to everybody
  - Needed for various protocols
    - e.g. Gnutella, OSPF, DHCP
- Link-layer
  - Layer-2, our last stop on our journey
  - What happens on link between hosts
- Error detection/correction
  - Parity
  - Checksum
  - CRC