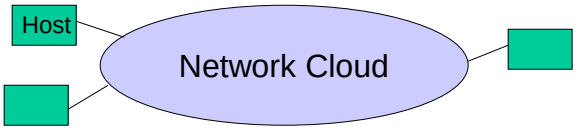
Introduction

EE 607

Sasaki

Internet Network Architecture

Network: big switch, transporting information end-to-end



Information: blocks (files) or streams of bits

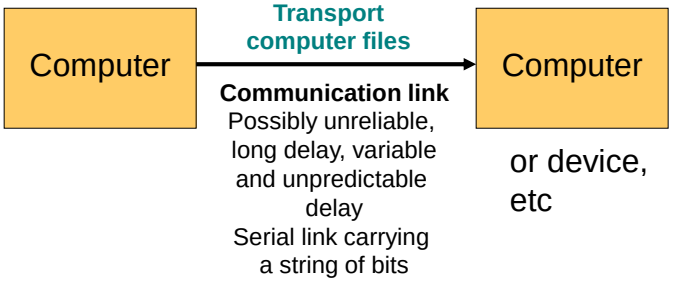
Packet-Switched Network: *packets* are transported

Packet: file of bits + control information

Overview

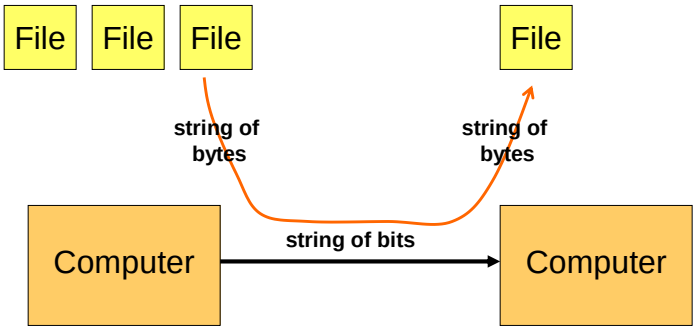
- Simple computer communication
- Simple computer networks
  - One-hop networks
- Overview of network architecture
  - Multi-hop networks
- Routing
- Read Stevens Chaps 1 and 2

Computer Communication



Computers may have local clocks with (almost) the same rate. Not synchronized.

Computer Communication

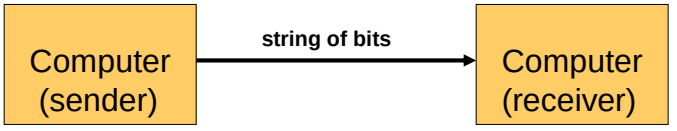


When does a file start and end?
What about errors in transmissions?
How do multiple applications share the link?

Overview

- Dealing with local unsynchronized clocks
- Delimiting file transfers
- Managing errors
  - Forward error correction
  - Error detection and retransmit
- Multiplexing transfers of different applications
- What's a packet?
- Implementation

Local Clocks

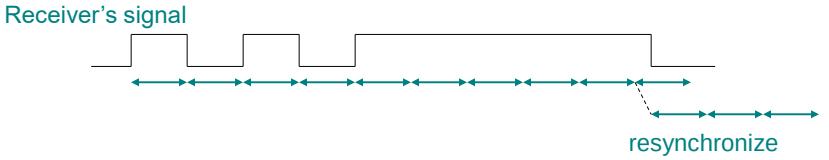


Sends a bit per clock period

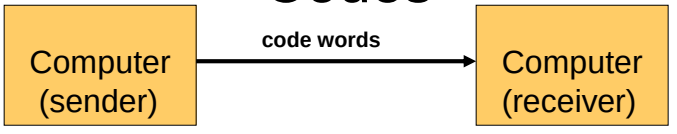
Sends "0" or "1" for a whole clock period

Every clock period it checks link and decides "0" or "1"

BUT clock drift (not exactly the same as sender)



One Approach: Block Line Codes



code book (table)

Data	Codeword
0000	01001
0001	01101
0010	10010
0011	01010
.	.
.	.
.	.
1111	00110

Distinct codewords, with nice transmission properties
e.g., enough transitions, DC balance

Example Block Line Code

DC Balance

Binary block line code

00000	01000	10000	11000
00001	01001	10001	11001
00010	01010	10010	11010
00011	01011	10011	11011
00100	01100	10100	11100
00101	01101	10101	11101
00110	01110	10110	11110
00111	01111	10111	11111

20 “balanced” code words

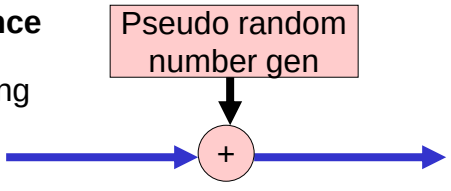
16 used for data (4 bits)

4 used for other things

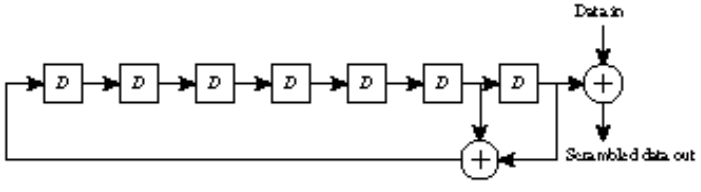
Modulation and Demodulation

DC Balance

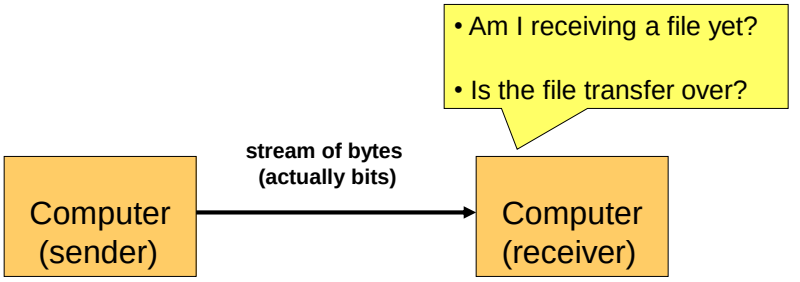
Scrambling



“randomized” output that avoids long strings of 0s or 1s



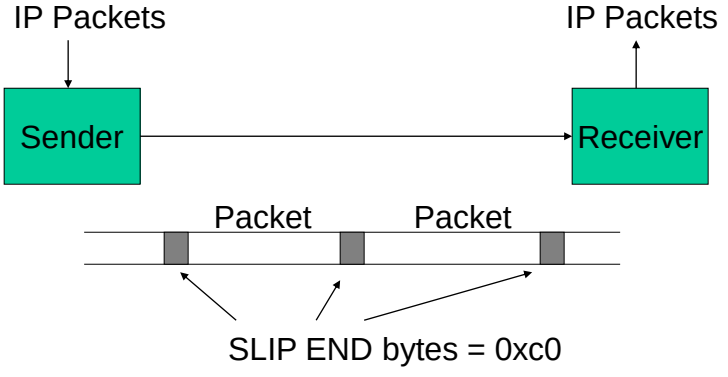
Delimiting Files



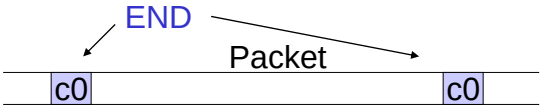
- Sequence of files and blanks ("idle fill").
- We can think of a blank as a "blank file".
- We need delimiters, e.g., "commas"

Serial Line IP (SLIP)

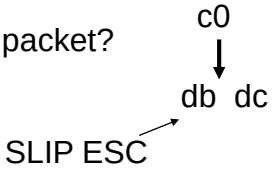
Packets = files for now -- we'll discuss later



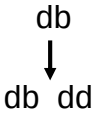
Serial Line IP (SLIP)



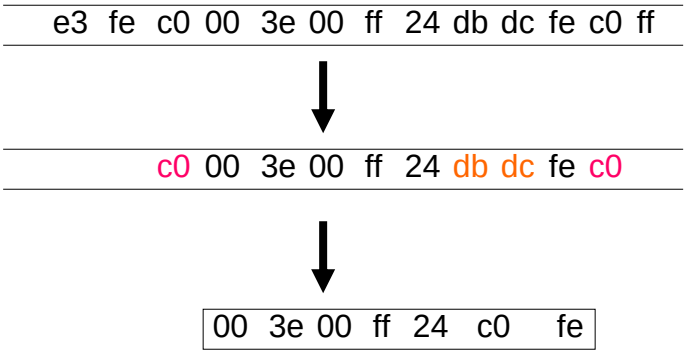
- What if there's a 0xc0 byte in the packet?



- What if there's a 0xdb byte in the packet?



Where is the packet?



Other Delimiting Schemes

- Bit stuffing
  - Example: every subsequence ..011111x.. stuff a “0” to get ..0111110x...
    - Unstuff at the receiver
- Header Error Check (HEC) based frame delineation
  - Some packet formats will cyclic redundancy checks on their headers
  - When synchronization is lost then keep doing CRC until getting a hit. Then assume it’s a header

Managing Errors

Forward Error Correction (FEC):

Data is encoded to correct for any loss of information

-- Error correcting codes

Automatic Repeat Request (ARQ):

Error detecting codes are used

Bad files are retransmitted

Error Detecting Codes

Simple Parity Bit: detect odd number of bit errors

Data string:	1010011
Even parity: even number of ones	1010011**0**
Odd parity: odd number of ones	1010011**1**

Even parity bit = XORing all bits in data string

Error Detection

Checksum: sum of a bunch of numbers
if a number changes then the check sum will detect it

A byte string: 2,3,5,15,9
Checksum: 34

Addition: ordinary or ones-complement, which is bit-wise XOR

Error Detection: Division

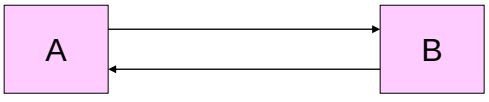
Cyclic Redundancy Checks (CRC):

Example: Code is defined by a prime #, e.g., 7

Data = 3946
Codeword = 3946X
Codeword should be divisible by 7
39469/7 --> Remainder = 3
39469-3 = 39466 is divisible by 7
Thus, X = 6

If codeword is changed then it is likely to be detected by dividing by 7 and checking remainder.

ARQ: Scenario

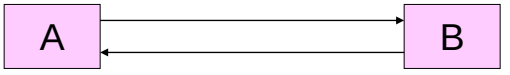


A sends files to B

B sends feedback information to A
e.g., ACK, received correctly
NACK, received in error

Link assumptions: errors are detected
packets and feedback may be lost

Simple ARQ



A
1. send file
2. wait for feedback
3. if ACK, go to 1
   else if NACK, resend
   and go to 2.

B
1. wait for file
2. if correct, send ACK
   else send NACK

Need time-outs. Even with timeouts it gets hairy.

Error Control

Error Control Codes

Possible Data Blocks	**Codewords**
000	001011
001	110101
010	001010
011	101101
100	010011
101	000111
110	010101
111	101010

Sending Side
Data Block:011

Codeword:101101

Transmission

Received: 100101

Decoding Decision
101101 was recvd
Data: 011

Simple FEC

Repetition Code

Data 0: Transmit 000
Data 1: Transmit 111

Majority vote detection

Even Parity Bit:

Extra bit so that a string of bits has even number of 1s

0 0 1 1 0 0 1 1

Data block

Error Correction

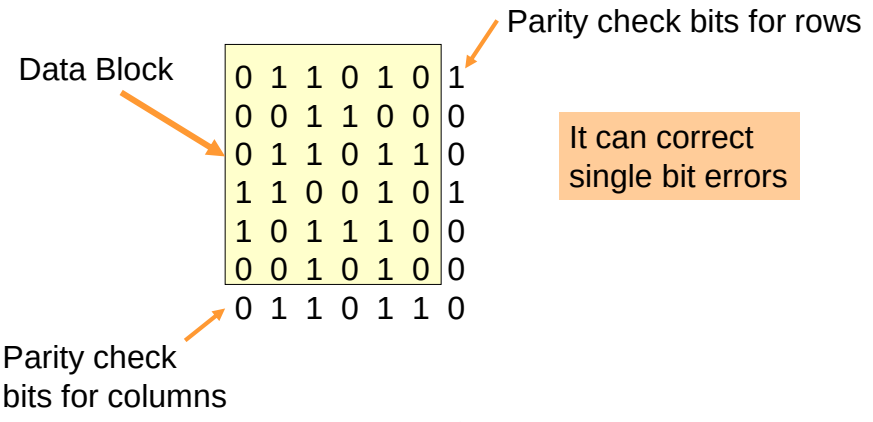
Horizontal-vertical parity check

0	1	0	1	1	0	1	0	**0**
1	1	1	0	0	1	1	1	**0**
0	1	1	0	1	0	0	1	**0**
0	0	1	1	0	1	0	0	**1**
0	1	1	1	0	0	1	1	**1**
1	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**

Single-bit error correction

Simple FEC

Horizontal-Vertical Parity Check



It can correct single bit errors

Reed-Solomon Codes

- Data Block is a string of *k* symbols (e.g., bytes)
- Extra *r* "redundant" symbols are added to make it a codeword
- Codeword has $n = k + r$ symbols
- Can correct up to *r/2* errors
- Restriction: $n = 2^m - 1$, where 2^m = number of symbols
- Restriction: *r* must be even
- Codes are defined by (n,k)
  - ITU-T Example: (255,239) -- 7% redundancy
  - ITU-T Example: (255,223) -- 15% redundancy

High density code

Reed-Solomon Codes

Analogy

Data: a0, a1
Polynomial: a1 x z + a0
Send five points on the line. Receiving any 3 will allow us to find a1 and a0

Reed-Solomon Codes

Performance Trade-Offs vs. Unencoded

Unencoded advantage
• Can send at lower rate and more energy per bit
by n/k = 1 + r/k
Less errors for transmitted bits
Coding advantage
• Can correct transmission errors

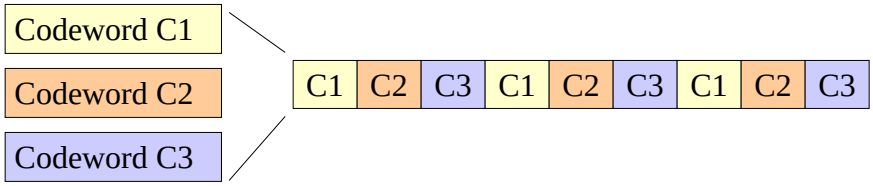
Coding gain: reduction in BER of data bits (not transmitted bits)

Burst Errors

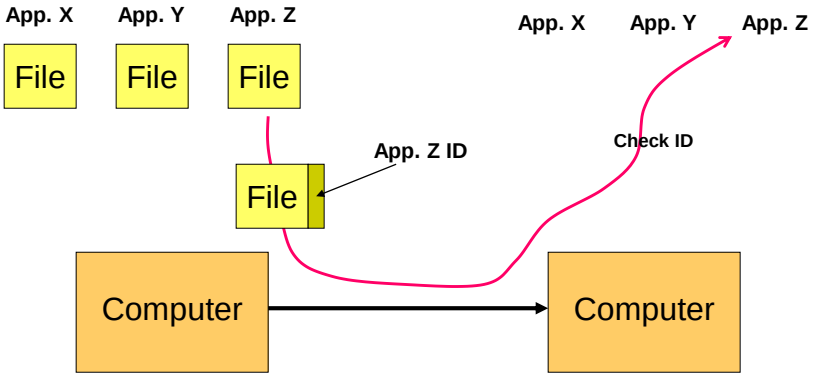
Reed-Solomon Codes

Can correct a burst of errors in *r/2* symbols
Can correct a burst of *m x* (*r/2*) bit errors

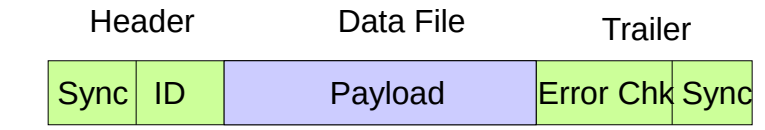
Interleaving can distribute a burst of errors to multiple codewords, producing correctable smaller bursts



Multiplexing for Multiple Applications



What's a Packet?
Data Files with Overhead

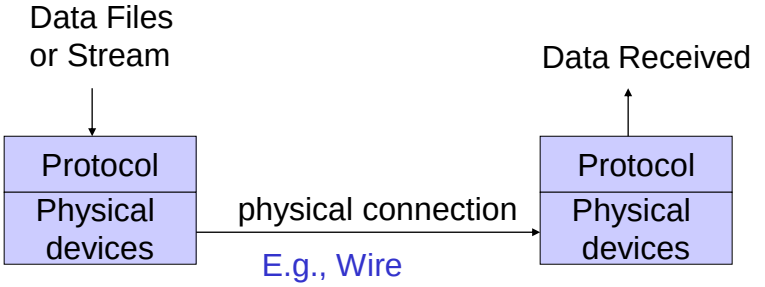


For multiplexing/demultiplexing

Delimiters ("commas")

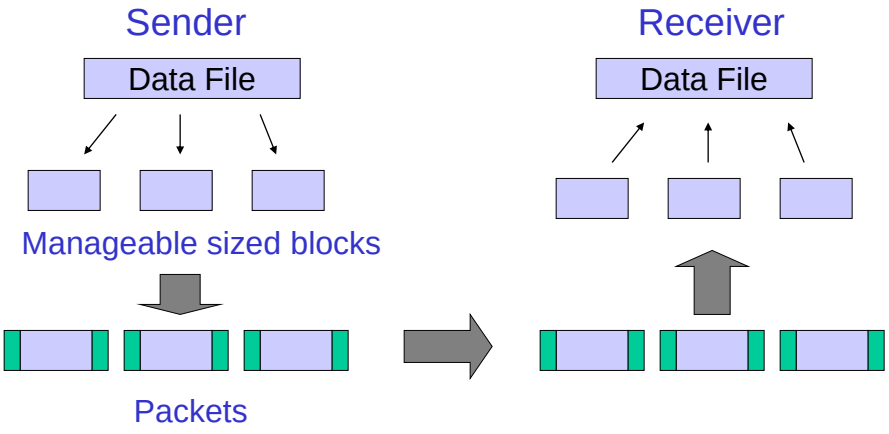
Other items
are in the header
but we don't have
time to discuss

Simple Computer Communication

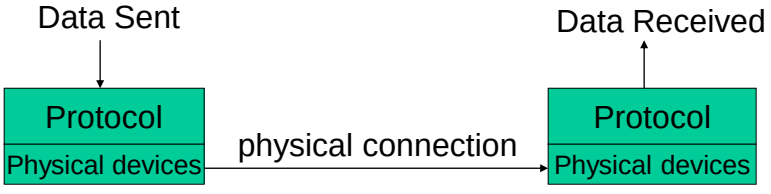


E.g., Wire
Radio
Optical fiber

Simple Computer Communication



Simple Computer Communication



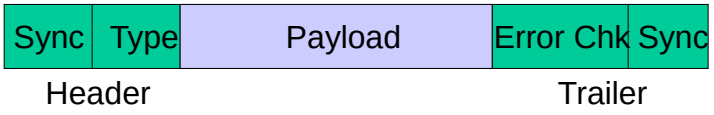
Sending Protocol

- **Break data into blocks**
- **Wrap blocks into packets**
- **Transmit packets**

Receiving Protocol

- **Receive packets**
- **Unwrap payload from packets**
- **Reconstruct data from payloads**

Example Simple Packet



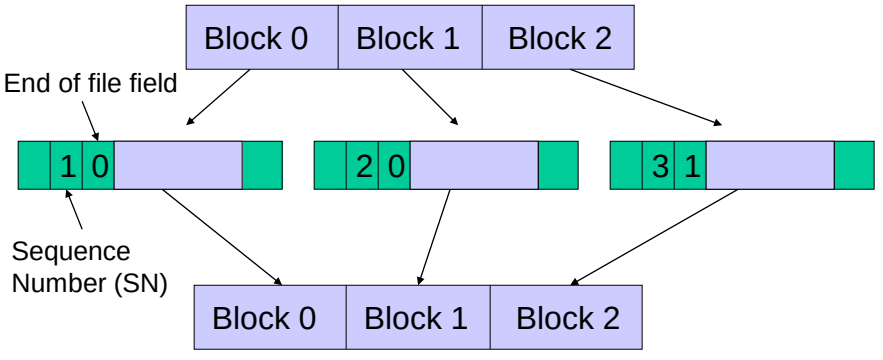
Payload = data to transport

Sync = Delimiter, e.g., “comma”.

Type = type of packet: e.g., data, control, type of protocol

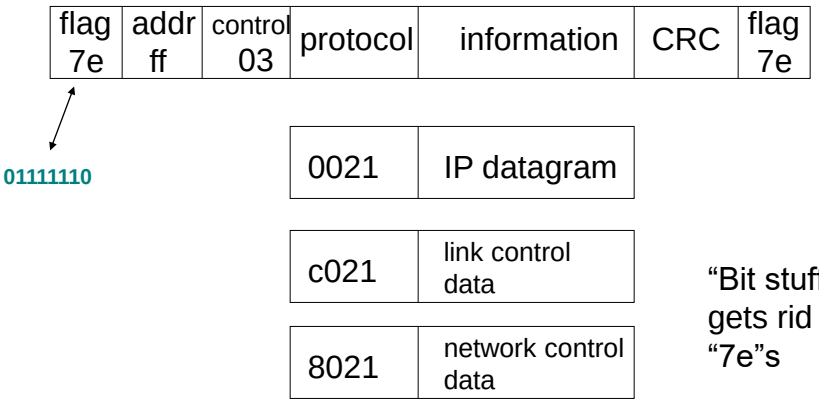
Error Check: for error detection e.g., parity bit, check sum, *cyclic redundancy check* (CRC)

Fragmentation/Reassembly



Packet Header May Have Sequence Number & EOF Fields

PPP: Point-to-Point Protocol



"Bit stuffing" gets rid of "7e"s

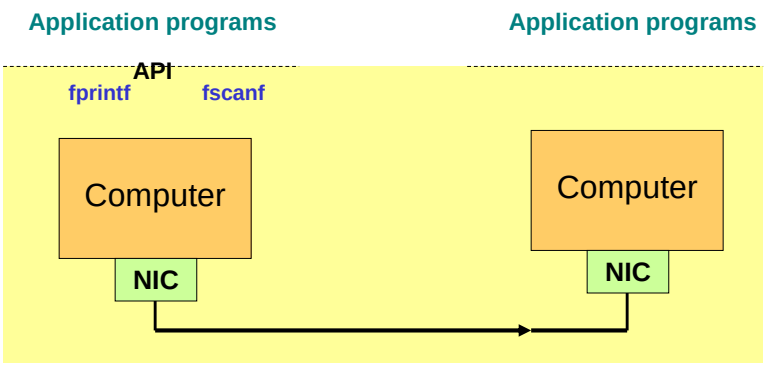
Implementation

Network Interface Card (NIC): the hardware
NIC is IO to the computer
- When NIC receives a packet, it interrupts the computer
- An interrupt handler deals with the NIC

Operating systems deals with the NIC

How Do I Develop an Application?

Application Programmer's Interface (API)

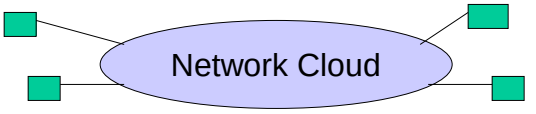


Simple Computer Networks

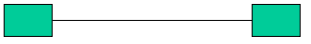
- One-hop networks
  - Ethernet
  - Token bus
  - Token ring
- Multi-hop networks

Computer Networks

Computer Network: allows multiple devices (> 2) to communicate

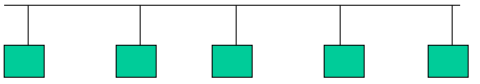


NOT A Network: point-to-point communication



Example Simple Network

Bus (multi-access or shared link)



Local Area Network (LAN)

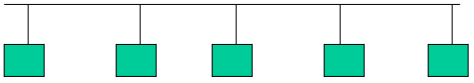
Broadcast link: everyone can hear everyone

New Issues:

(1) which packet is for who?
(2) contention

Example Simple Network

Bus (multi-access or shared link)



(1) Which packet is for who?
- Addresses: each node (host) has a unique address/ID
- Packet header: destination address field
  source address field (not necessary)

(2) Contention: Access control
I.e., who gets to transmit on the bus?

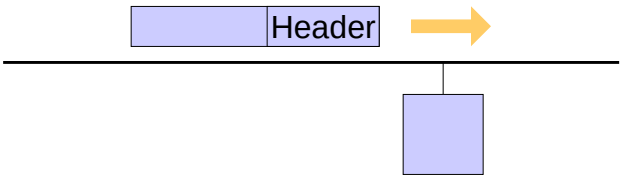
Media Access Control

Who gets access to the broadcast link?

Carrier-Sense Multiple-Access/Collision Detection (CSMA/CD):
E.g., Ethernet

At a node:
- Always listens to the bus for packets
- Stores packets that are destined for it

CSMA/CD



- Copy packet
- Check header
- Discard if packet is for another node

CSMA/CD: Transmitting

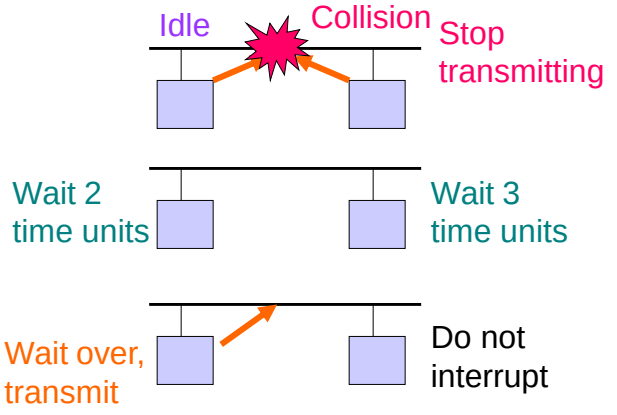
At a node with a packet to transmit:

- Wait until bus is idle -- [*carrier sense*]
- Transmit packet
- Listen while transmitting, and if there's garbage
  (due to another node transmitting), stop transmitting.
  Wait a random delay, then retransmit.
  [*collision detection*]

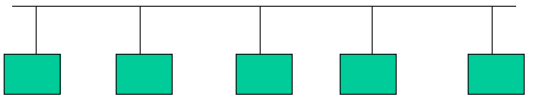
CSMA/CD: Transmitting

Both transmit at the same time

Random wait



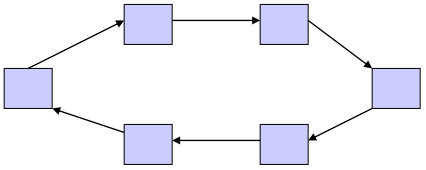
Another Example: Token Bus



Everybody takes turns at transmitting in round-robin fashion

A token (small control message) is passed around from one node to the next.
Whoever has the token gets to transmit.
When it's done it passes it along.

Token Ring

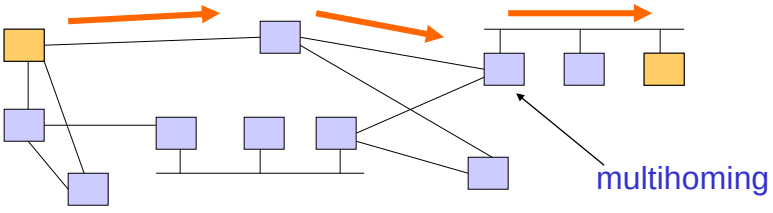


Nodes normally just forward what they receive:
ring behaves as a broadcast link.

There's a single token being passed around.

When a node has a packet, it holds onto token and then transmits packet: It releases token when it's done.

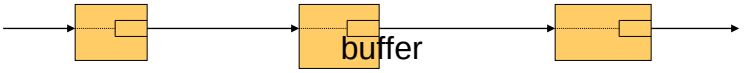
Multi-Hop Network



Multi-hop network: it may take more than one link to get between nodes.

Routing: finding a path for a packet from source to destination.

Store-and-Forwarding



Packets must be completely received at a node before they can be forwarded on the next link.

Other Concepts

- MTU: Maximum transmission unit
  - Path MTU: MTU along a path
- Loopback interface

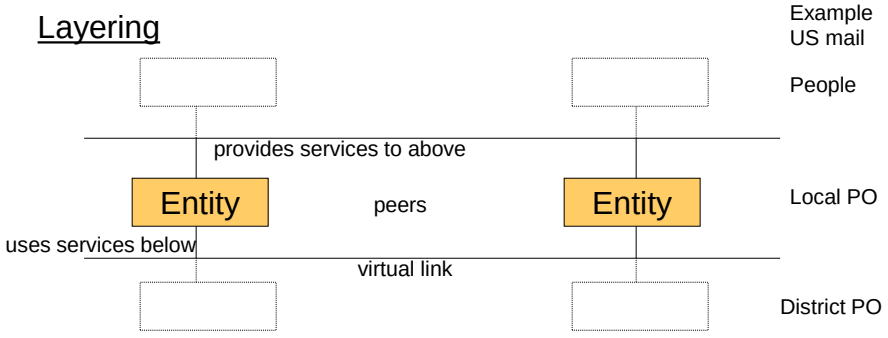
Network Architecture

- **Layering concept**
- **TCP/IP**
- **OSI**

Layering -- Network Architecture

Modularity/Hierarchy: managing complexity

Layering



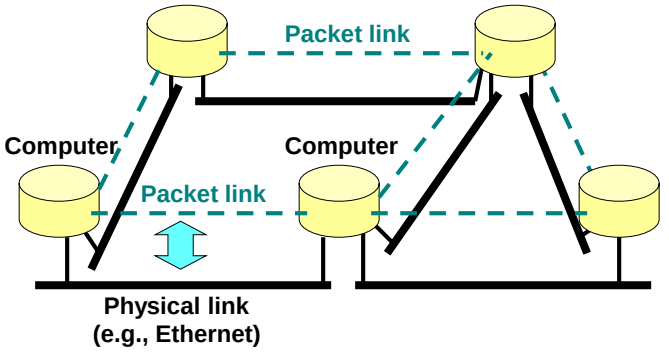
TCP/IP Protocol Suite

Application	Telnet, FTP, e-mail, etc
Transport	TCP, UDP (end-to-end transport from user point of view)
Network	IP, ICMP, IGMP (end-to-end transport from network point of view, routing)
Link	Device driver and interface card

Link Layer (Bottom)

Link layer makes physical links transfer packets

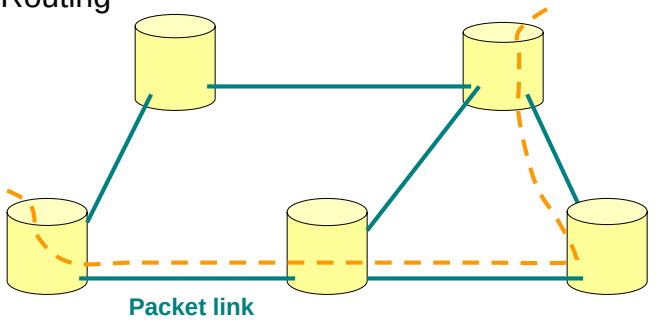
Packet links = physical links + link layer protocol



Next Layer: IP

IP Layer provides end-to-end transport but unreliable
"best effort delivery"

Routing



TCP Layer (Above IP)

TCP: Uses IP transport service and provides
reliable end-to-end transport

TCP: Deals with losses, reordering, and delay.

The Four Layers

Application Layer: handles the details of particular application.

Transport Layer: provides a flow of data between two hosts, for the application above.

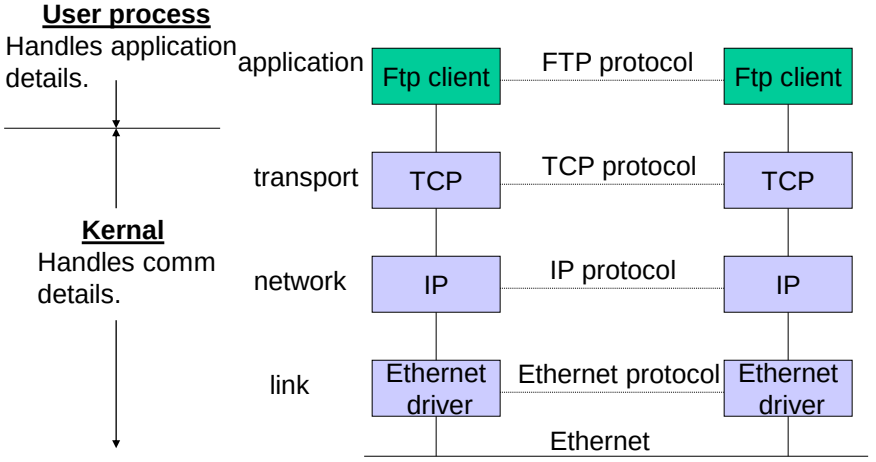
TCP: reliable end-to-end flow with packets in sequence. Supports large files or streams of files/bits.

UDP: transports single packets (called datagrams). Useful for short messages or e-mail.

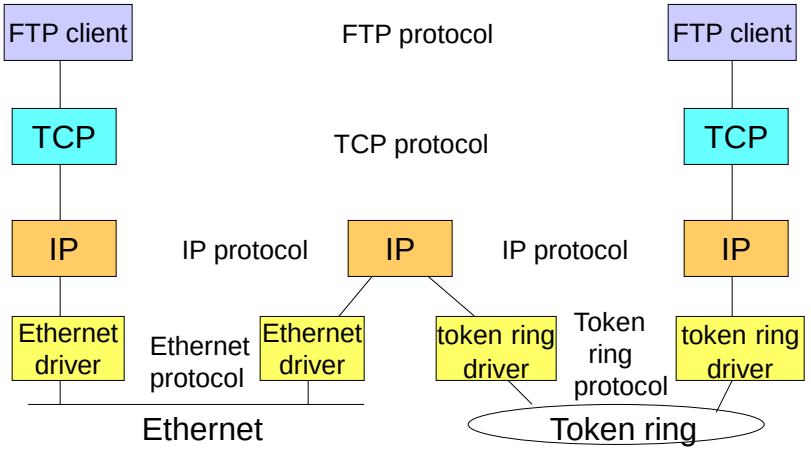
Network Layer: handles movement of packets in the network. Routing is done here.

Link Layer: handles the hardware details of physical transmission

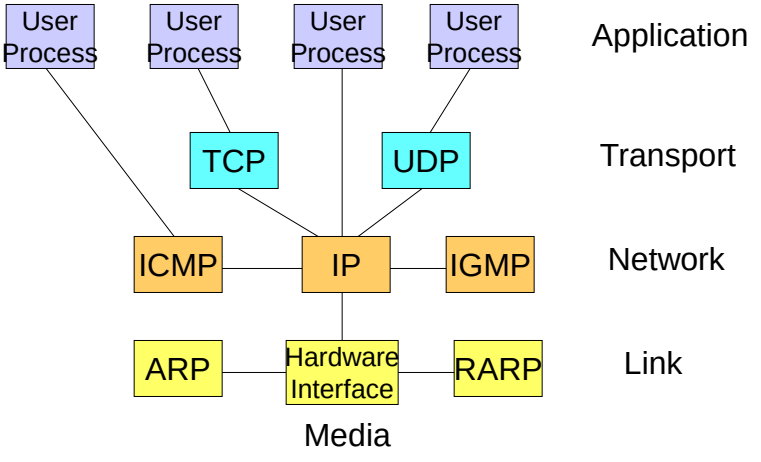
Two Hosts on a LAN running FTP



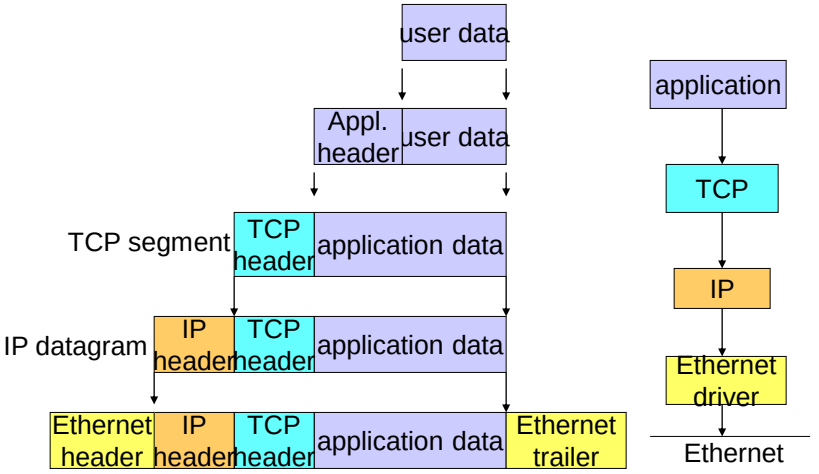
Two networks connected with a router



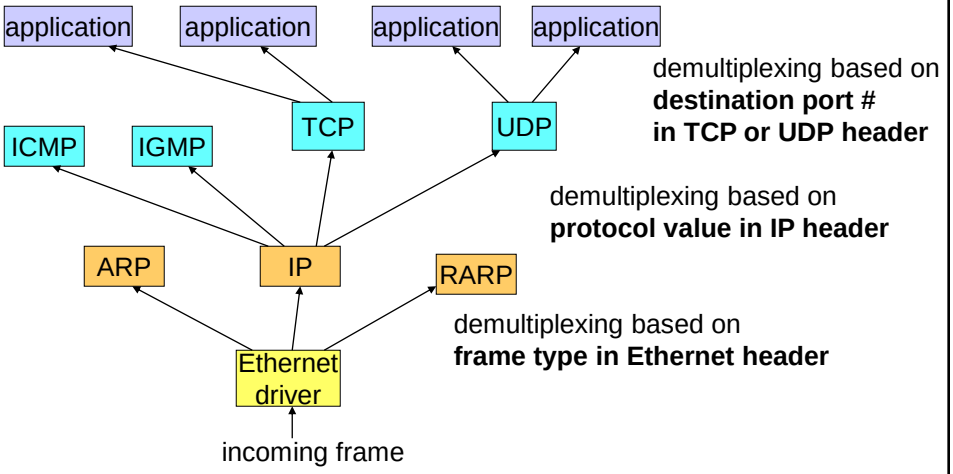
TCP/IP Protocol Suite



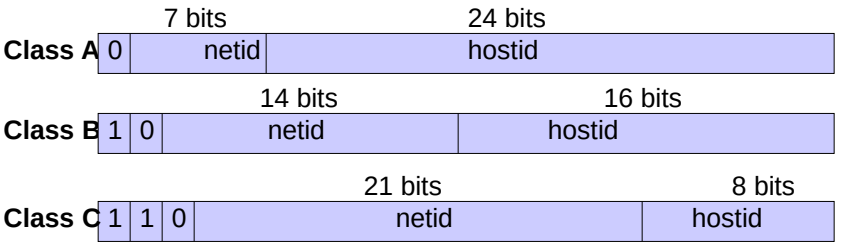
Encapsulation



Demultiplexing



IP Address



Class D: Multicasting. Class E: Future use

Class	Range
A	0.0.0.0 to 127.255.255.255
B	128.0.0.0 to 191.255.255.255
C	192.0.0.0 to 223.255.255.255

Addresses are for interfaces

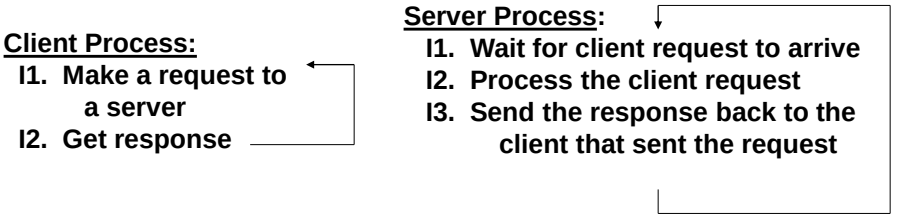
Domain Name System

DNS: Distributed database that does mapping between IP addresses and host names

spectra.eng.hawaii.edu = 128.171.61.59

Applications

Client-Server Model



Application Programming Interfaces (API)

"Sockets"

Applications

Application Programming Interfaces (APIs):

Sockets

Application programs communicate by reading and writing to virtual files

fscanf(fp,…) fprintf(fp,….)

Port Numbers: ties UDP or TCP frames to processes

Standards

Internet Society (ISOC)

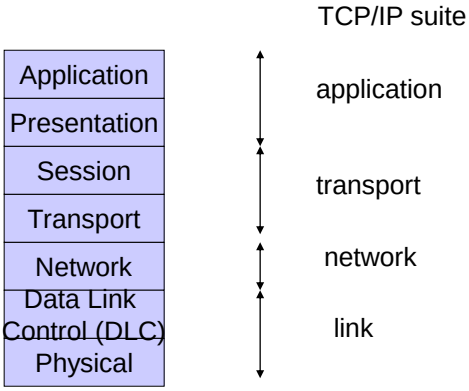
Internet Architecture Board (IAB)

Internet Engineering Task Force (IETF) www.ietf.org

Request For Comments (RFCs)

OSI Architecture

Another important architecture, but we won't get into it.



Routing

- **Packet switched routing**
- **Connectionless**
- **Connection-oriented**
- **Source routing**

Packet Switched Routing

- Routing: moving packets through the network
- Route: path that a packet follows
  - Connectionless
  - Connection-oriented
  - Source routing

Connectionless (Datagram) Routing

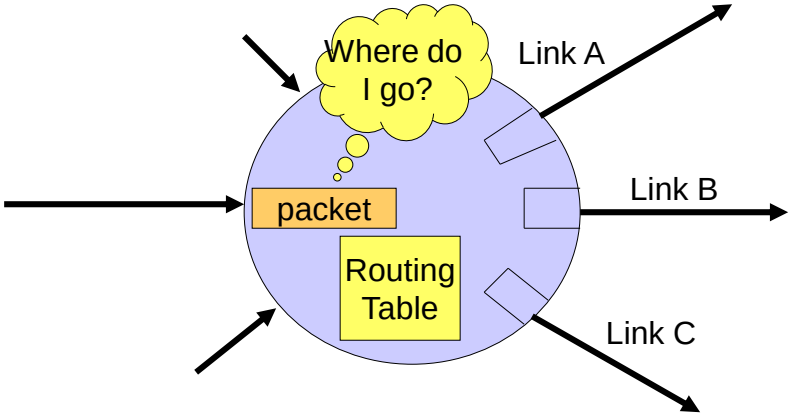
- Each packet has the destination address

Dest addr	

- Each node has a routing table

Dest addr	Outgoing link

Connectionless (Datagram) Routing



Connectionless (Datagram) Routing

Routing Table:

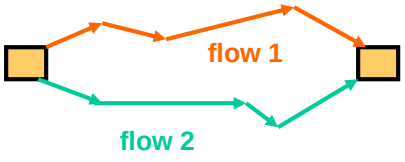
Destination	Outgoing Link
wiliki.eng.hawaii.edu	A
nsf.gov	B
ece.ucsd.edu	B
www.yahoo.com	C
darpa.mil	C
starbulletin.com	A

- Packet finds its own way. No set-up.
- Routing information in the packet header is the dest. addr.

Connection-Oriented outing (Virtual Circuit Switched)

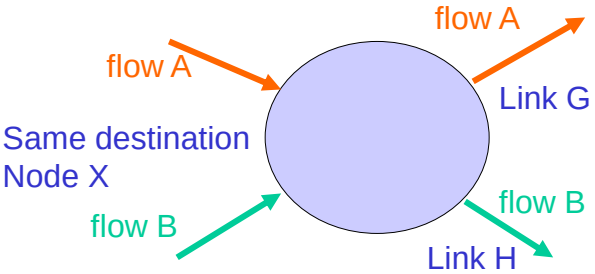
Packets are in "flows" or "streams":

- There may be multiple flows between pairs of end nodes.
- Each flow has its own route,
  i.e., it is "pinned" to the route.



Connection-Oriented Routing

Can't use destination address in packet.



Ambiguity in forwarding

Dest	Out link
X	G
X	H

Connection-Oriented Routing

Packet has "label" or "tag" that identifies the flow.

Label	

Routing tables have tags.

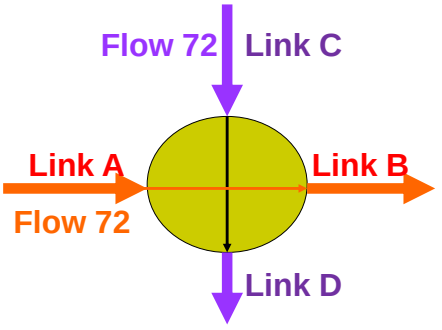
Tags (label)	Outgoing Links
flow A	G
flow B	H

Connection management

- Flow set-up: add flow to routing routing tables along the path
- Flow termination: delete flow from routing tables

More On Labels

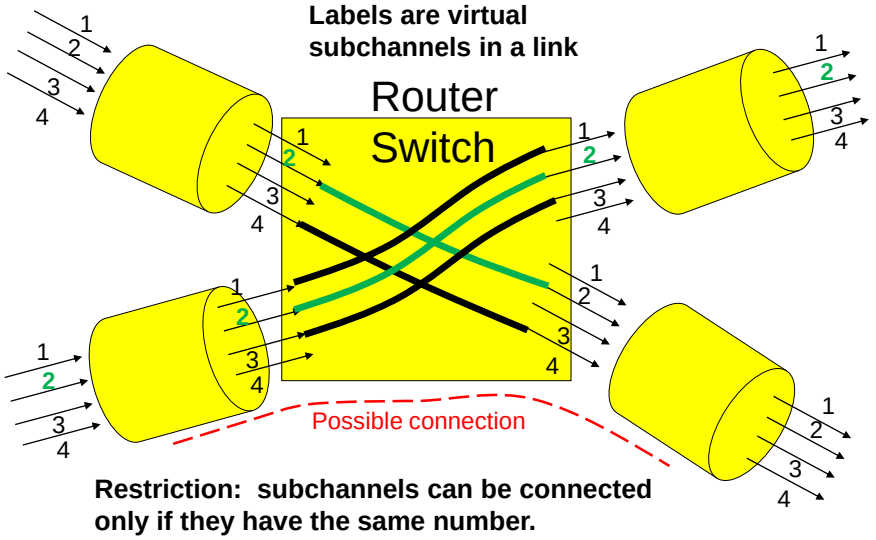
Multiple flows can use the same label
through a node.
But they must use different links
Label / Link = ID for a connection



Routing Table

Incoming	Outgoing
A/72 C/72	B D

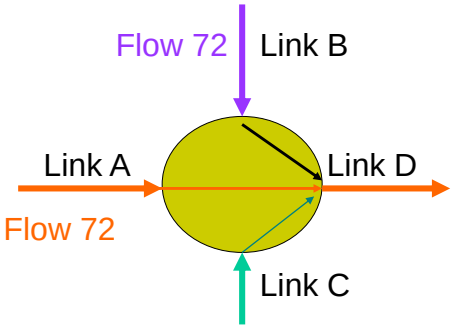
Another Perspective



Restriction: subchannels can be connected only if they have the same number.

More On Labels: Label Swapping

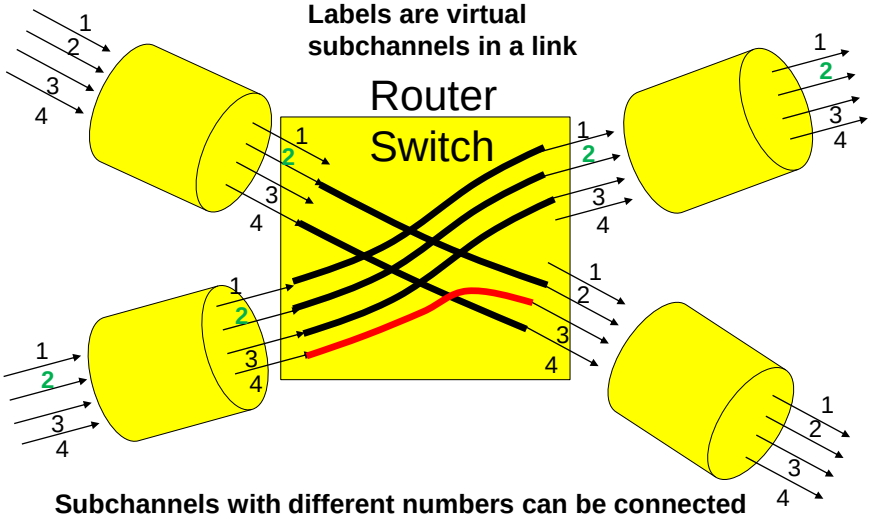
Label swapping: packets change labels along the network.



Routing Table

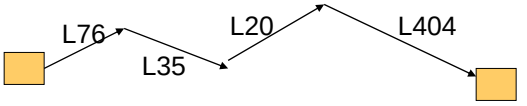
Incoming	Outgoing
A/72	D/72
B/72	D/73
C/72	D/74

Another Perspective



Source Routing

Routes of packets are known at their source nodes



Route is stored in packet header

packet header

No tables are needed.

Types of Packets

Unicast: single destination -- most packets are of this type.

Broadcast: *every* node is a destination -- useful for control messages.
Intermediate nodes must be able to duplicate packets.

Multicast: multiple destinations, but not all nodes. Group communication, etc.