CS 640: Introduction to Computer Networks

Aditya Akella

Lecture 1
Introduction

http://www.cs.wisc.edu/~akella/CS640/F07

1

Goals of This Class

- Understand principles and practice of networking
- How are modern networks designed? Operated? Managed?
- Performance and design trade-offs in network protocols and applications
- How do network applications work? How to write applications that use the network?
  - Hands-on approach to understand network internals
- How will different aspects of networking evolve in the future?

2

Goal of Networking

- Enable *communication* between *network applications* on different *end-points*
  - End-points? computers, cell phones….
  - Application? Web, Peer to Peer, Streaming video, IM
  - Communication? transfer bits or information across a "network"
- Network must understand application needs/demands
  - What data rate?
  - Traffic pattern? (bursty or constant bit rate)
  - Traffic target? (multipoint or single destination, mobile or fixed)
  - App sensitivity? (to delay, "jitter", loss)
  - Difficulty: Network may not know these in the first place!
- How does the application "use" the network?
  - Peer to peer: how to find nearest host
  - Web: how to modulate sending rate? Coexist with other users/apps?

3

Defining a "Network"

- Network = nodes + links
  - Will build on this soon
- Intentionally vague. There are several different networks:
  - The Internet
  - Wisc CS network
  - Telephone network
  - Home wireless networks
  - Others – sensor nets, "On Star", cellular networks
- Our focus on Internet
  - Also explore important common issues and challenges

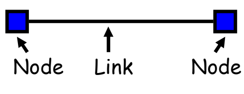
4

Challenges for Networking

- Accommodate different geographic scopes
  - The Internet vs. home network
- Enable scale
  - CS network vs. the Internet
- Seamlessly integrate different application types
  - Email vs. video conferencing
- Independent administration and Trust
  - Corporate network – owned by one entity
  - Internet owned and managed by 17,000 network providers
    - Independent, conflicting interests

5

Network Building Block: Links

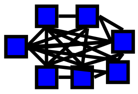


- "Physical"-layer questions
  - Wired or wireless
  - Voltage (Electrical) or wavelength (optical)
- "Link"-layer issues: How to send data?
  - Medium access – can either side talk at once?
  - Data format?

6

Basic Building Block: Links

- … But what if we want more hosts?



Wires for everybody?

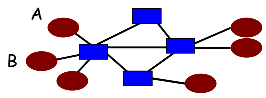
How many wires?

- How many additional wires per host?
- Scalability?

7

Key Idea: Multiplexing

- Multiplex: share network resources
  - Resources need "provisioning"
  - Grow at slower rate than number of nodes

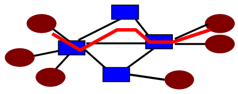


- How to share? Switched network
  - Party "A" gets resources sometimes
  - Party "B" gets them sometimes
- Interior nodes act as "Switches"

8

Circuit Switching

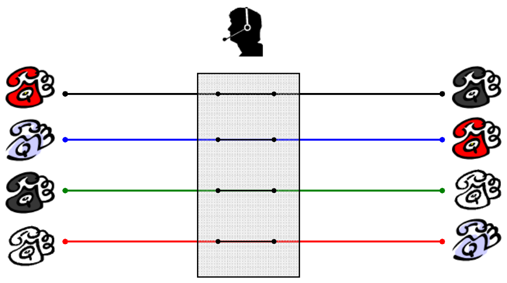
- Source first establishes a circuit to destination
  - Switches along the way stores info about connection
    - Possibly allocate resources
    - Different srs-dst's get different paths



- Source sends the data over the circuit
  - No address required since path is established beforehand
- The connection is explicitly set up and torn down
- Switches use TDM (digital) or FDM (analog) to transmit data from various circuits

9

Switching in the Telephone Network



10

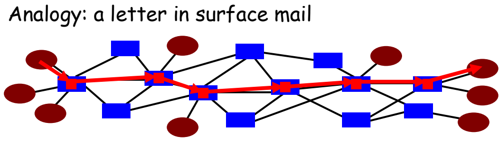
Circuit Switching Discussion

- Positives
  - *Fast and simple* data transfer, once the circuit has been established
  - Predictable performance since the circuit provides *isolation* from other users
    - E.g. guaranteed max bandwidth
- Negatives
  - How about bursty traffic
    - Circuit will be idle for significant periods of time
    - Also, can't send more than max rate
  - Circuit set-up/tear down is expensive
  - Also, reconfiguration is slow
    - Fast becoming a non-issue

11

Packet Switching

- Source sends information as self-contained packets
  - Packets have an address.
  - Source may have to break up single message in multiple packets
- Packets travel independently to the destination host
  - Switches use the address in the packet to determine how to forward the packets
  - "Store and forward"
- Analogy: a letter in surface mail

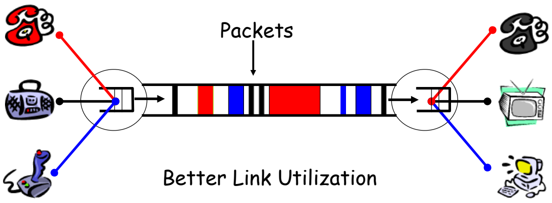


12

Benefits of Statistical Multiplexing

TDM: Flow gets chance in fixed time-slots

SM: Flow gets chance on demand; no need to wait for slot



13

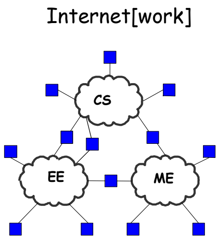
Packets vs. Circuits

- Efficient
  - Can send from any input that is ready
  - No notion of wastage of resources that could be used otherwise
- Contention (i.e. no isolation)
  - Congestion
  - Delay
- Accommodates bursty traffic
  - But need packet buffers
- Address look-up and forwarding
  - Need optimization
- Packet switching pre-dominant
  - Circuit switching used on large time-scales, low granularities

14

Internetwork

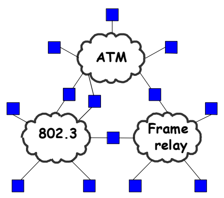
- A collection of interconnected networks
- **Networks**: Different depts, labs, etc.
- **Router**: node that connects distinct networks
- **Host**: network endpoints (computer, PDA, light switch, …)
- Together, an independently administered entity
  - Enterprise, ISP, etc.



15

Internetwork Challenges

- Many differences between networks
  - Address formats
  - Performance – bandwidth/latency
  - Packet size
  - Loss rate/pattern/handling
  - Routing
- How to translate and inter-operate?
  - Routers are key to many of these issues



16

"The Internet"

- Internet vs. internet
- The Internet: the interconnected set of networks of the Internet Service Providers (ISPs) and end-networks, providing data communications services.
  - Network of internetworks, and more
  - About 17,000 different ISP networks make up the Internet
  - Many other "end" networks
  - 100,000,000s of hosts

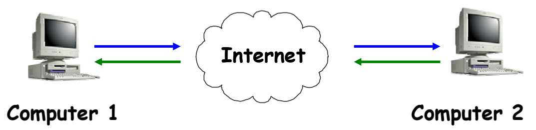
17

Internet Design Issues

- Extra Slides…
  - We will cover these topics in greater detail in future lectures

18

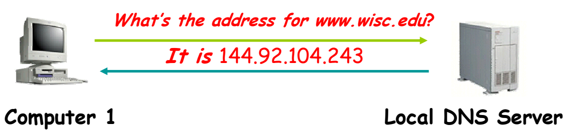
Some Key "Internet" Design Issues



Need:
(1) naming,
(2) addressing and
(3) routing
(4) …

19

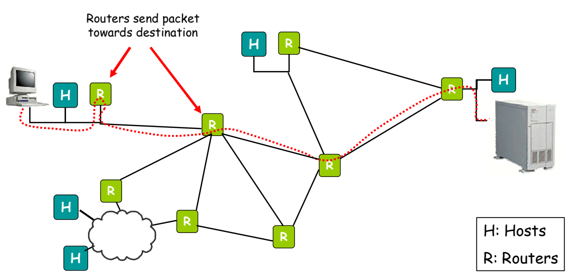
Key Issues: Naming/Addressing



Translates human readable names to logical endpoints

20

Key Issues: Routing



21

Key Issues: Network Service Model

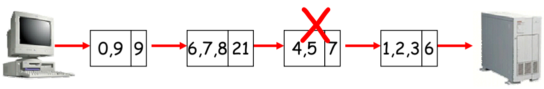
- What is the *service model?*
  - Defines what to expect from the network
  - ***Best-effort:*** packets can get lost, no guaranteed delivery
- What if you want more?
  - Performance guarantees (QoS)
  - Reliability
    - Corruption
    - Lost packets
  - In-order delivery for file chunks
  - Etc…

22

What if the Data gets Corrupted?

Problem: Data Corruption

Solution: Add a *checksum*



23

What if the Data gets Lost?

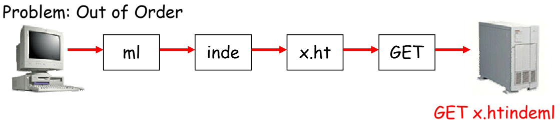
Problem: Lost Data

Solution: Timeout and Retransmit

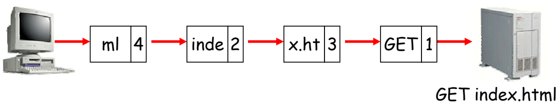
24

What if Data is Out of Order?

Problem: Out of Order



Solution: Add Sequence Numbers



25

Meeting Application Demands

- Sometimes network can do it
  - E.g., Quality of Service
    - Benefits of circuit switching in packet-switched net
    - Hard in the Internet, easy in restricted contexts
    - Lecture 20
- OR hosts can do it
  - E.g., end-to-end *Transport protocols*
    - TCP performs end-to-end retransmission of lost packets to give the illusion of a reliable underlying network.
    - Lectures 16-19

26

To Summarize…

Networks implement many functions

- Links
- Sharing/Multiplexing
- Routing
- Addressing/naming
- Reliability
- Flow control
- Fragmentation
- Etc….

27

CS 640: Introduction to Computer Networks

Aditya Akella

Lecture 2
Layering, Protocol Stacks, and Standards

1

Today's Lecture

- Layers and Protocols
- A bit about applications

2

Network Communication: Lots of Functions Needed

- Links
- Multiplexing
- Routing
- Addressing/naming (locating peers)
- Reliability
- Flow control
- Fragmentation

How do you implement these functions?
Key: Layering and protocols

3

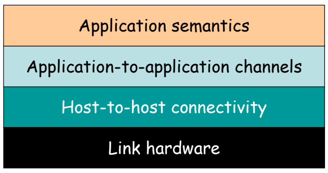
What is Layering?

- A way to deal with complexity
  - Add multiple levels of abstraction
  - Each level encapsulates some key functionality
  - And exports an interface to other components
  - Example?
- Layering: Modular approach to implementing network functionality by introducing abstractions
- Challenge: how to come up with the "right" abstractions?

4

Example of Layering

- Software and hardware for communication between two hosts

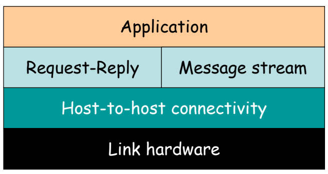


- Advantages:
  - Simplifies design and implementation
  - Easy to modify/evolve

5

What is a Protocol?

- Could be multiple abstractions at a given level
  - Build on the same lower level
  - But provide diferent service to higher layers
- Protocol: Abstract object or module in layered structure



6

1. Protocols Offer Interfaces

- Each protocol offers interfaces
  - One to higher-level protocols on the same end hosts
    - Expects one from the layers on which it builds
    - Interface characteristics, e.g. IP service model
  - A "peer interface" to a counterpart on destinations
    - Syntax and semantics of communications
    - (Assumptions about) data formats
- Protocols build upon each other
  - Adds value, improves functionality overall
    - E.g., a reliable protocol running on top of IP
  - Reuse, avoid re-writing
    - E.g., OS provides TCP, so apps don't have to rewrite

7

2. Protocols Necessary for Interoperability

- Protocols are the key to interoperability.
  - Networks are very heterogenous:

Ethernet: 3com, etc.	Hardware/link
Routers: cisco, juniper etc.	Network
App: Email, AIM, IE etc.	Application

  - The hardware/software of communicating parties are often not built by the same vendor
  - Yet they can communicate because they use the same protocol
    - Actually implementations could be different
    - But must adhere to same specification
- Protocols exist at many levels.
  - Application level protocols
  - Protocols at the hardware level

8

OSI Model

- One of the first standards for layering: OSI
- Breaks up network functionality into seven layers
- This is a "reference model"
  - For ease of thinking and implementation
- A different model, TCP/IP, used in practice

9

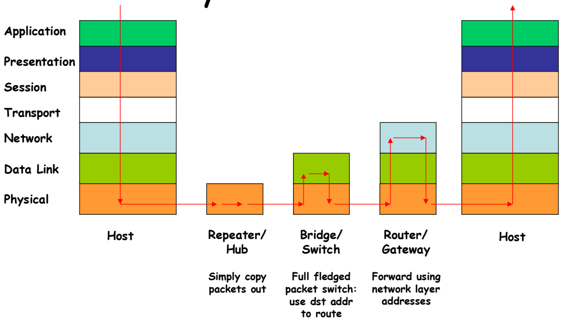
The OSI Standard: 7 Layers

1. **Physical**: transmit bits (link)
2. **Data link**: collect bits into frames and transmit frames (adaptor/device driver)
3. **Network**: route packets in a packet switched network
4. **Transport**: send messages across processes end2end
5. **Session**: tie related flows together
6. **Presentation**: format of app data (byte ordering, video format)
7. **Application**: application protocols (e.g. FTP)

- OSI very successful at shaping thought
- TCP/IP standard has been amazingly successful, and it's not based on a rigid OSI model

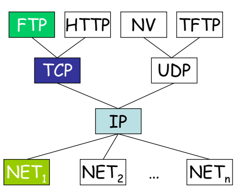
10

OSI Layers and Locations



11

The Reality: TCP/IP Model



App protocols

Two transport protocols: provide logical channels to apps

Interconnection of n/w technologies into a single logical n/w

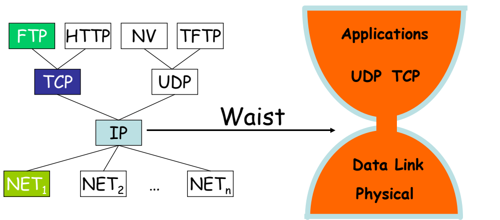
Network protocols implemented by a comb of hw and sw.

Note: No strict layering.

App writers can define apps that run on any lower level protocols.

12

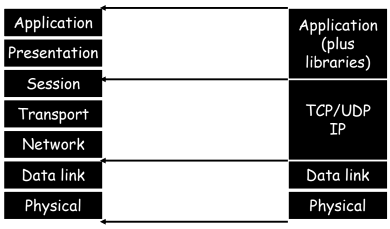
The Thin Waist



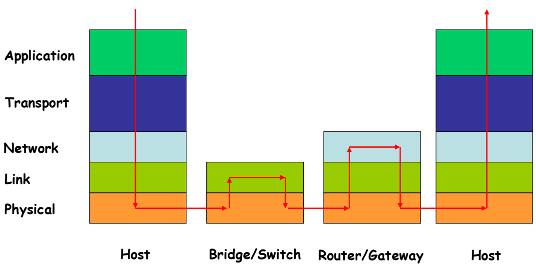
The Hourglass Model

The waist: minimal, carefully chosen functions.
Facilitates interoperability and rapid evolution

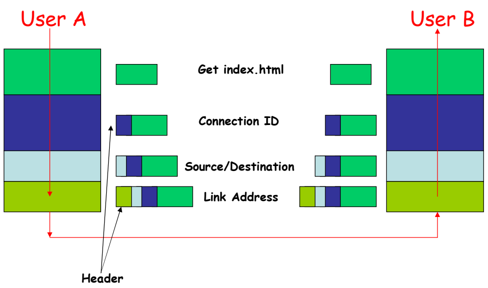
TCP/IP vs OSI



TCP/IP Layering



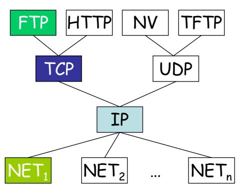
Layers & Encapsulation

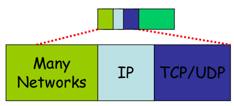


16

Protocol Demultiplexing

- Multiple choices at each layer
- How to know which one to pick?

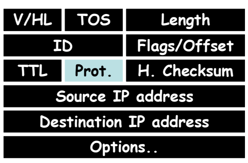


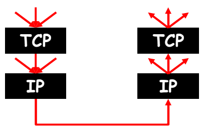


17

Multiplexing & Demultiplexing

- Multiple implementations of each layer
  - How does the receiver know what version/module of a layer to use?
- Packet header includes a demultiplexing field
  - Used to identify the right module for next layer
  - Filled in by the sender
  - Used by the receiver
- Multiplexing occurs at multiple layers. E.g., IP, TCP, …





18

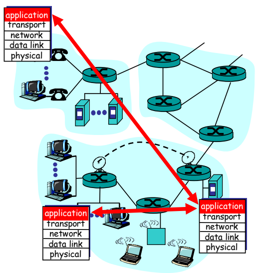
Layering vs Not

- Layer N may duplicate layer N-1 functionality
  - E.g., error recovery
- Layers may need same info (timestamp, MTU)
- Strict adherence to layering may hurt performance
- Some layers are not always cleanly separated
  - Inter-layer dependencies in implementations for performance reasons
  - Many cross-layer assumptions, e.g. buffer management
- Layer interfaces are not really standardized.
  - It would be hard to mix and match layers from independent implementations, e.g., windows network apps on unix (w/o compatibility library)

19

Applications; Application-Layer Protocols

- Application: communicating, distributed processes
  - Running in network hosts in "user space"
  - N/w functionality in kernel space
  - Exchange messages to implement app
  - e.g., email, file transfer, the Web
- Application-layer protocols
  - One "piece" of an app
  - Define messages exchanged by apps and actions taken
  - Use services provided by lower layer protocols



20

Writing Applications: Some Design Choices

- Communication model:
  - Client-server or peer-to-peer
  - Depends on economic and usage models
- Transport service to use?
  - "TCP" vs "UDP"
  - Depends on application requirements

21

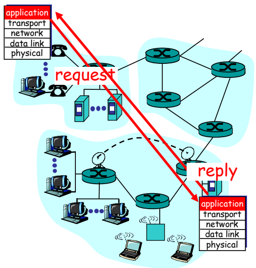
Client-Server Paradigm vs. P2P

Typical network app has two pieces: *client* and *server*

Client:
- Initiates contact with server ("speaks first")
- Typically requests service from server,
- For Web, client is implemented in browser; for e-mail, in mail reader

Server:
- Provides requested service to client
- e.g., Web server sends requested Web page, mail server delivers e-mail

- P2P is a very different model
  - *No* notion of client or server



22

Choosing the Transport Service

Data loss
- Some applications (e.g., audio) can tolerate some loss
- Other applications (e.g., file transfer, telnet) require 100% reliable data transfer

Timing
- Some applications (e.g., Internet telephony, interactive games) require low delay to be "effective"

Bandwidth
- Some applications (e.g., multimedia) require a minimum amount of bandwidth to be "effective"
- Other applications ("elastic apps") will make use of whatever bandwidth they get

23

Transmission Control Protocol (TCP)

TCP
- Reliable – guarantee delivery
- Byte stream – in-order delivery
- Connection-oriented – single socket per connection
- Setup connection followed by data transfer

Telephone Call
- Guaranteed delivery
- In-order delivery
- Connection-oriented
- Setup connection followed by conversation

Example TCP applications
Web, Email, Telnet

24

User Datagram Protocol (UDP)

UDP
- No guarantee of delivery
- Not necessarily in-order delivery
- Datagram – independent packets; connectionless
- Must address each packet

Postal Mail
- Unreliable
- Not necessarily in-order delivery
- Letters sent independently
- Must address each reply

Example UDP applications
Multimedia, voice over IP

25

Transport Service Requirements of Common Applications

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
web documents	no loss	elastic	no
real-time audio/ video	loss-tolerant	audio: 5Kb-1Mb video:10Kb-5Mb	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few Kbps	yes, 100's msec
financial apps	no loss	elastic	yes and no

26

CS 640: Computer Networking

Yu-Chi Lai

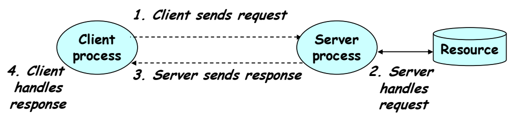
Lecture 3
Network Programming

Topics

- Client-server model
- Sockets interface
- Socket primitives
- Example code for echoclient and echoserver
- Debugging With GDB
- Programming Assignment 1 (MNS)

Client/server model

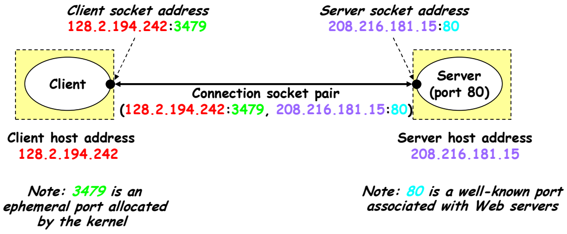
- Client asks (*request*) – server provides (*response*)
- Typically: single server - multiple clients
- The server does not need to know *anything* about the client
  - even that it exists
- The client should always know *something* about the server
  - at least where it is located



Note: clients and servers are processes running on hosts (can be the same or different hosts).

Internet Connections (TCP/IP)

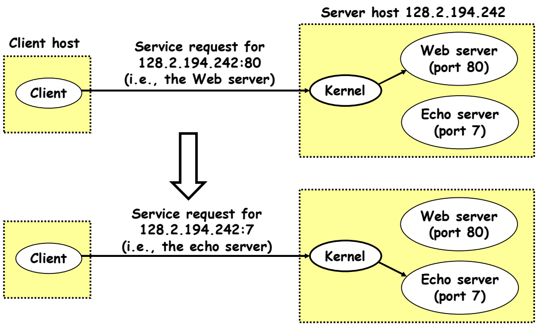
- Address the machine on the network
  - By IP address
- Address the process
  - By the "port"-number
- The pair of *IP-address + port* – makes up a "*socket-address*"



Clients

- Examples of client programs
  - Web browsers, `ftp`, `telnet`, `ssh`
- How does a client find the server?
  - The IP address in the server socket address identifies the host
  - The (well-known) port in the server socket address identifies the service, and thus implicitly identifies the server process that performs that service.
  - Examples of well known ports
    - Port 7: Echo server
    - Port 23: Telnet server
    - Port 25: Mail server
    - Port 80: Web server

Using Ports to Identify Services

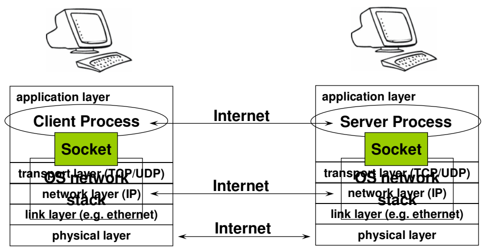


Servers

- Servers are long-running processes (daemons).
  - Created at boot-time (typically) by the init process (process 1)
  - Run continuously until the machine is turned off.
- Each server waits for requests to arrive on a well-known port associated with a particular service.
  - Port 7: echo server
  - Port 23: telnet server
  - Port 25: mail server
  - Port 80: HTTP server
- Other applications should choose between 1024 and 65535

See /etc/services for a comprehensive list of the services available on a Linux machine.

Sockets as means for inter-process communication (IPC)



The interface that the OS provides to its networking subsystem

Sockets

- What is a socket?
  - To the kernel, a socket is an endpoint of communication.
  - To an application, a socket is a file descriptor that lets the application read/write from/to the network.
    - Remember: All Unix I/O devices, including networks, are modeled as files.
- Clients and servers communicate with each by reading from and writing to socket descriptors.
- The main distinction between regular file I/O and socket I/O is how the application "opens" the socket descriptors.

Socket Programming Cliches

- Network Byte Ordering
  - Network is big-endian, host may be big- or little-endian
  - Functions work on 16-bit (short) and 32-bit (long) values
  - htons() / htonl() : convert host byte order to network byte order
  - ntohs() / ntohl(): convert network byte order to host byte order
  - Use these to convert network addresses, ports, …

```
struct sockaddr_in serveraddr;
/* fill in serveraddr with an address */
…
/* Connect takes (struct sockaddr *) as its second argument */
connect(clientfd, (struct sockaddr *) &serveraddr,
        sizeof(serveraddr));
…
```

- Structure Casts
  - You will see a lot of 'structure casts'

Socket primitives

- **SOCKET: int socket(int *domain*, int *type*, int *protocol*);**
  - *domain* := AF_INET (IPv4 protocol)
  - *type* := (SOCK_DGRAM *or* SOCK_STREAM)
  - *protocol* := 0 (IPPROTO_UDP *or* IPPROTO_TCP)
  - *returned*: socket descriptor (*sockfd*), -1 is an error

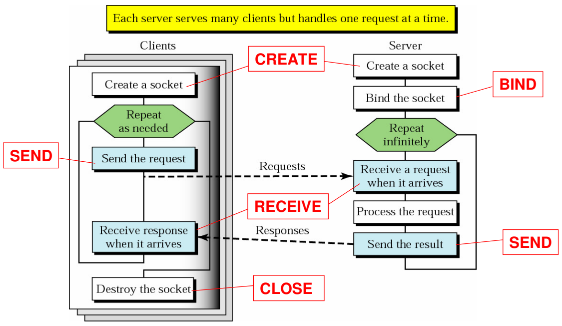
- **BIND: int bind(int *sockfd*, struct sockaddr **my_addr*, int *addrlen*);**
  - *sockfd* - socket descriptor (returned from socket())
  - *my_addr:* socket address, struct sockaddr_in is used
  - *addrlen* := sizeof(struct sockaddr)

```
struct sockaddr_in  {
  unsigned short  sin_family;  /* address family (always AF_INET) */
  unsigned short  sin_port;    /* port num in network byte order */
  struct in_addr  sin_addr;    /* IP addr in network byte order */
  unsigned char   sin_zero[8]; /* pad to sizeof(struct sockaddr) */
};
```

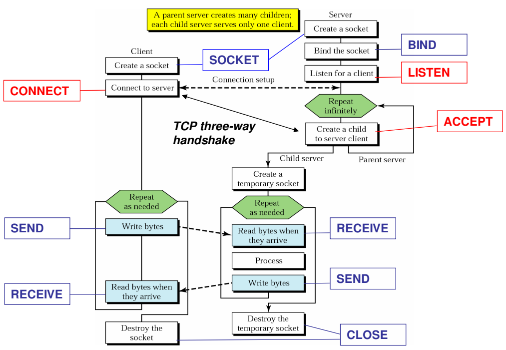
- **LISTEN: int listen(int *sockfd*, int *backlog*);**
  - *backlog*: how many connections we want to queue
- **ACCEPT: int accept(int *sockfd*, void **addr*, int **addrlen*);**
  - *addr*: here the socket-address of the caller will be written
  - *returned:* a new socket descriptor (for the temporal socket)
- **CONNECT: int connect(int *sockfd*, struct sockaddr **serv_addr*, int *addrlen*); //used by TCP client**
  - parameters are same as for bind()
- **SEND: int send(int *sockfd*, const void **msg*, int *len*, int *flags*);**
  - *msg*: message you want to send
  - *len:* length of the message
  - *flags* := 0
  - *returned:* the number of bytes actually sent
- **RECEIVE: int recv(int *sockfd*, void **buf*, int *len*, unsigned int *flags*);**
  - *buf:* buffer to receive the message
  - *len:* length of the buffer ("don't give me more!")
  - *flags* := 0
  - *returned:* the number of bytes received

- **SEND** (DGRAM-style): **int sendto(int *sockfd*, const void **msg*, int *len*, int *flags*, const struct sockaddr **to*, int *tolen*);**
  - *msg*: message you want to send
  - *len*: length of the message
  - *flags* := 0
  - *to*: socket address of the remote process
  - *tolen*: = sizeof(struct sockaddr)
  - *returned*: the number of bytes actually sent
- **RECEIVE** (DGRAM-style): **int recvfrom(int *sockfd*, void **buf*, int *len*, unsigned int *flags*, struct sockaddr **from*, int **fromlen*);**
  - *buf*: buffer to receive the message
  - *len*: length of the buffer ("don't give me more!")
  - *from*: socket address of the process that sent the data
  - *fromlen*:= sizeof(struct sockaddr)
  - *flags* := 0
  - *returned*: the number of bytes received
- **CLOSE**: **close (*socketfd*);**

Client+server: connectionless

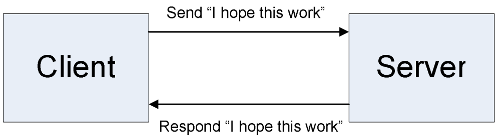


Client+server: connection-oriented



Concurrent server

Echo Client-Server



#include’s

```
#include <stdio.h>      /* for printf() and fprintf() */
#include <sys/socket.h> /* for socket(), connect(),
                           sendto(), and recvfrom() */
#include <arpa/inet.h>  /* for sockaddr_in and
                           inet_addr() */
#include <stdlib.h>     /* for atoi() and exit() */
#include <string.h>     /* for memset() */
#include <unistd.h>     /* for close() */
#include <netdb.h>      /* Transform the ip address
                           string to real uint_32 */

#define ECHOMAX 255     /* Longest string to echo */
```

EchoClient.cpp -variable declarations

```
int main(int argc, char *argv[])
{
   int sock;                    /* Socket descriptor */
   struct sockaddr_in echoServAddr; /* Echo server address */
   struct sockaddr_in fromAddr;   /* Source address of echo */
   unsigned short echoServPort =2000;   /* Echo server port */
   unsigned int fromSize;         /* address size for recvfrom() */
   char *servIP=“172.24.23.4”;   /* IP address of server */
   char *echoString=“I hope this works”;   /* String to send to
   echo server */
   char echoBuffer[ECHOMAX+1];     /* Buffer for receiving
   echoed string */
   int echoStringLen;            /* Length of string to echo */
   int respStringLen;            /* Length of received response */
```

EchoClient.c - creating the socket

```
/* Create a datagram/UDP socket  and
   error check */
sock = socket(AF_INET, SOCK_DGRAM,
   0);
if(sock <= 0){
    printf("Socket open error\n");
   exit(1);
  }
```

EchoClient.cpp – sending

```
/* Construct the server address structure */
memset(&echoServAddr, 0, sizeof(echoServAddr)); /* Zero out
    structure */
echoServAddr.sin_family = AF_INET; /* Internet addr family */
inet_pton(AF_INET, servIP, &echoServAddr.sin_addr); /* Server IP
    address */
echoServAddr.sin_port = htons(echoServPort); /* Server port */

/* Send the string to the server */
echoStringLen = strlen(echoString);
 sendto(sock, echoString, echoStringLen, 0, (struct sockaddr *)
    &echoServAddr, sizeof(echoServAddr);
```

EchoClient.cpp – receiving and printing

```
/* Recv a response */
fromSize = sizeof(fromAddr);
recvfrom(sock, echoBuffer, ECHOMAX, 0, (struct sockaddr *)
    &fromAddr, &fromSize);

/* Error checks like packet is received from the same server*/
…

/* null-terminate the received data */
echoBuffer[echoStringLen] = '\0';
printf("Received: %s\n", echoBuffer); /* Print the echoed arg */
close(sock);
exit(0);
} /* end of main () */
```

EchoServer.cpp – creating socket

```
int main(int argc, char *argv[])
{
   int sock;                   /* Socket */
   struct sockaddr_in echoServAddr; /* Local address */
   struct sockaddr_in echoClntAddr; /* Client address */
   unsigned int cliAddrLen;       /* Length of incoming message */
   char echoBuffer[ECHOMAX];       /* Buffer for echo string */
   unsigned short echoServPort =2000;    /* Server port */
   int recvMsgSize;                /* Size of received message */

  /* Create socket for sending/receiving datagrams */
  sock = socket(AF_INET, SOCK_DGRAM, 0);
  if(sock <= 0){
      printf("Socket open error\n");
      exit(1);
 }
```

EchoServer.cpp – binding

```
/* Construct local address structure*/
   memset(&echoServAddr, 0, sizeof(echoServAddr));  /* Zero out structure
  */
  echoServAddr.sin_family = AF_INET; /* Internet address family */
  echoServAddr.sin_addr.s_addr =htonl(INADDR_ANY);
  echoServAddr.sin_port = htons((uint16_t) echoServPort); /* Local port */

  /* Bind to the local address */
 int error_test = bind(sock, (struct sockaddr *) &echoServAddr,
  sizeof(echoServAddr));
 if(error_test < 0){
    printf("Binding error\n");
    exit(1);
 }
```

EchoServer.cpp – receiving and echoing

```
for (;;) /* Run forever */
  {
     cliAddrLen = sizeof(echoClntAddr);

     /* Block until receive message from a client */
     recvMsgSize = recvfrom(sock, echoBuffer, ECHOMAX, 0,
        (struct sockaddr *) &echoClntAddr, &cliAddrLen);

     printf("Handling client %s\n", inet_ntoa(echoClntAddr.sin_addr));

     /* Send received datagram back to the client */
     sendto(sock, echoBuffer, recvMsgSize, 0,
         (struct sockaddr *) &echoClntAddr, sizeof(echoClntAddr));
  }

} /* end of main () */
```

Error handling is must

Socket Programming Help

- man is your friend
  - man accept
  - man sendto
  - Etc.
- The manual page will tell you:
  - What #include<> directives you need at the top of your source code
  - The type of each argument
  - The possible return values
  - The possible errors (in errno)

Debugging with gdb

- Prepare program for debugging
  - Compile with "-g" (keep full symbol table)
  - Don't use compiler optimization ("-O", "-O2", …)
- Two main ways to run gdb
  - On program directly
    - `gdb progname`
    - Once gdb is executing we can execute the program with:
      - `run args`
  - On a core (post-mortem)
    - `gdb progname core`
    - Useful for examining program state at the point of crash
- Extensive in-program documentation exists
  - `help` (or `help <topic>` or `help <command>`)

More information…

- Socket programming
  - W. Richard Stevens, UNIX Network Programming
  - Infinite number of online resources
  - http://www.cs.rpi.edu/courses/sysprog/sockets/sock.html
- GDB
  - Official GDB homepage: http://www.gnu.org/software/gdb/gdb.html
  - GDB primer: http://www.cs.pitt.edu/~mosse/gdb-note.html

Project Partners

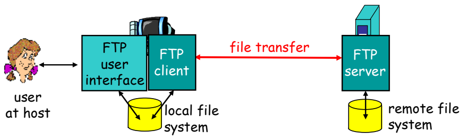
- If you don't have a partner
  - Stay back after class
- Now…
  - Overview of PA 1

CS640: Introduction to Computer Networks

Aditya Akella

Lecture 4 -
Application Protocols, Performance

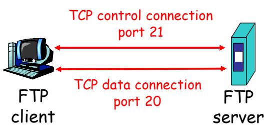
Applications
FTP: The File Transfer Protocol



- Transfer file to/from remote host
- Client/server model
  - *Client:* side that initiates transfer (either to/from remote)
  - *Server:* remote host
- ftp: RFC 959
- ftp server: port 21

FTP: Separate Control, Data Connections

- Ftp client contacts ftp server at port 21, specifying TCP as transport protocol
- Two parallel TCP connections opened:
  - Control: exchange commands, responses between client, server.
    "out of band control"
  - Data: file data to/from server



- Server opens data connection to client
  - Exactly one TCP connection per file requested.
  - Closed at end of file
  - New file requested → open a new data connection
- Ftp server maintains "state": current directory, earlier authentication

HTTP Basics

- HTTP layered over bidirectional byte stream
  - Almost always TCP
- Interaction
  - Client sends request to server, followed by response from server to client
  - Requests/responses are encoded in text
- Contrast with FTP
  - Stateless
    - Server maintains no information about past client requests
      - There are some caveats
  - In-band control
    - No separate TCP connections for data and control

Typical HTTP Workload (Web Pages)

- Multiple (typically small) objects per page
  - Each object a separate HTTP session/TCP connection
- File sizes
  - Why different than request sizes?
  - Heavy-tailed (both request and file sizes)
    - "Pareto" distribution for tail
    - "Lognormal" for body of distribution

Non-Persistent HTTP

http://www.cs.wisc.edu/index.html

1. Client initiates TCP connection
2. Client sends HTTP request for index.html
3. Server receives request, retrieves object, sends out HTTP response
4. Server closes TCP connection
5. Client parses index.html, finds references to 10 JPEGs
6. Repeat steps 1—4 for each JPEG
   (can do these in parallel)

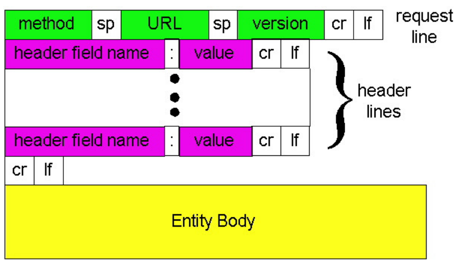
Issues with Non-Persistent HTTP

- Two "round-trip times" per object
  - RTT will be defined soon
- Server and client must maintain state per connection
  - Bad for server
  - Brand new TCP connection per object
- TCP has issues starting up ("slow start")
  - Each object face to face these performance issues
- HTTP/1.0

The Persistent HTTP Solution

- Server leaves TCP connection open after first response
  - W/O pipelining: client issues request only after previous request served
    - Still incur 1 RTT delay
  - W/ pipelining: client sends multiple requests back to back
    - Issue requests as soon as a reference seen
    - Server sends responses back to back
      - One RTT for all objects!
- HTTP/1.1

HTTP Request



HTTP Request

- Request line
  - Method
    - GET – return URI
    - HEAD – return headers only of GET response
    - POST – send data to the server (forms, etc.)
  - URL
    - E.g. /index.html if no proxy
    - E.g. http://www.cs.cmu.edu/~akella/index.html with a proxy
  - HTTP version

HTTP Request

- Request header fields
  - Authorization – authentication info
  - Acceptable document types/encodings
  - From – user email
  - If-Modified-Since
  - Referrer – what caused this page to be requested
  - User-Agent – client software
- Blank-line
- Body

HTTP Request Example

GET /~akella/index.html HTTP/1.1
Host: www.cs.wisc.edu
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
Connection: Keep-Alive

HTTP Response

- Status-line
  - HTTP version
  - 3 digit response code
    - 1XX – informational
    - 2XX – success
      - 200 OK
    - 3XX – redirection
      - 301 Moved Permanently
      - 303 Moved Temporarily
      - 304 Not Modified
    - 4XX – client error
      - 404 Not Found
    - 5XX – server error
      - 505 HTTP Version Not Supported
  - Reason phrase

HTTP Response

- Headers
  - Location – for redirection
  - Server – server software
  - WWW-Authenticate – request for authentication
  - Allow – list of methods supported (get, head, etc)
  - Content-Encoding – E.g x-gzip
  - Content-Length
  - Content-Type
  - Expires
  - Last-Modified
- Blank-line
- Body

HTTP Response Example

HTTP/1.1 200 OK
Date: Thu, 14 Sep 2006 03:49:38 GMT
Server: Apache/1.3.33 (Unix) mod_perl/1.29 PHP/4.3.10 mod_ssl/2.8.22 OpenSSL/0.9.7e-fips
Last-Modified: Tue, 12 Sep 2006 20:43:04 GMT
ETag: "62901bbe-161b-45071bd8"
Accept-Ranges: bytes
Content-Length: 5659
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html

<data data data>

Cookies: Keeping "state"

Many major Web sites use cookies
→ keep track of users
→ Also for convenience: personalization, passwords etc.

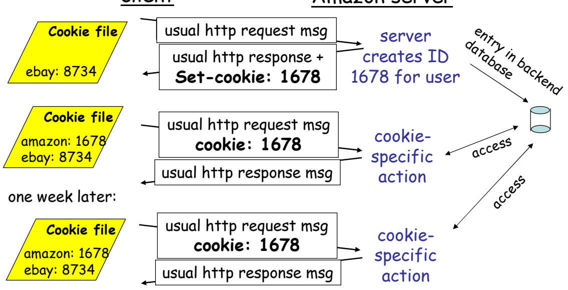
Four components:

1) Cookie header line in the HTTP response message
2) Cookie header line in HTTP request message
3) Cookie file kept on user's host and managed by user's browser
4) Back-end database at Web site

Example:

- Susan accesses Internet always from same PC
- She visits a specific e-commerce site for the first time
- When initial HTTP requests arrives at site, site creates a unique ID and creates an entry in backend database for ID

Cookies: Keeping "State" (Cont.)



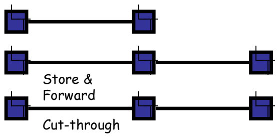
Performance Measures

- Latency or delay
  - How long does it take a bit to traverse the network
- Bandwidth
  - How many bits can be crammed over the network in one second?
- Delay-bandwidth product as a measure of capacity

Packet Delay: One Way and Round Trip

- Sum of a number of different delay components.
- Propagation delay on each link.
  - Proportional to the length of the link
- Transmission delay on each link.
  - Proportional to the packet size and 1/link speed
- Processing delay on each router.
  - Depends on the speed of the router
- Queuing delay on each router.
  - Depends on the traffic load and queue size
- This is one-way delay
  - Round trip time (RTT) = sum of these delays on forward and reverse path

Ignoring processing and queuing…



Prop + xmit

2*(Prop + xmit)

2*prop + xmit

Aside: When does cut-through matter?

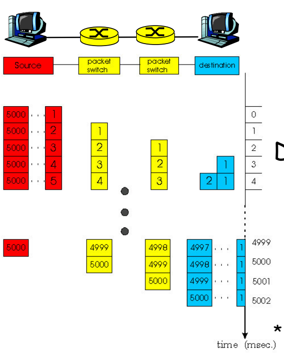
Routers have finite speed (processing delay)

Routers may buffer packets (queueing delay)

Ignoring processing and queuing…

Delay of one packet

Average sustained throughput



$\text{Delay}^* + \frac{\text{Size}}{\text{Throughput}}$

Units: seconds + bits/(bits/seconds)

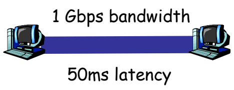
* For first bit to arrive

Some Examples

- How long does it take to send a 100 Kbit file? 10Kbit file?

Latency \ Throughput	100 Kbit/s	1 Mbit/s	100 Mbit/s
500 μsec	0.1005	0.0105	0.0006
10 msec	0.11	0.02	0.0101
100 msec	0.2	0.11	0.1001

Bandwidth-Delay Product



- Product of bandwidth and delay (duh!)
  - What is it above?
- What does this indicate?
  - #bytes sender can xmit before first byte reaches receiver
  - Amount of "in flight data"
- Another view point
  - B-D product == "capacity" of network from the sending applications point of view
  - Bw-delay amount of data "in flight" at all time → network "fully" utilized

TCP's view of BW-delay product

- TCP expects receiver to acknowledge receipt of packets
- Sender can keep up to RTT * BW bytes outstanding
  - Assuming full duplex link
  - When no losses:
    - 0.5RTT * BW bytes "in flight", unacknowledged
    - 05RTT * BW bytes acknowledges, acks "in flight"

Extra slides

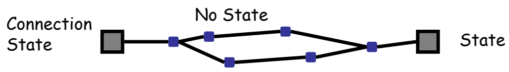
Internet Architecture

- Background
  - "The Design Philosophy of the DARPA Internet Protocols" (David Clark, 1988).
- Fundamental goal: "Effective techniques for multiplexed utilization of existing interconnected networks"
- "Effective" → sub-goals; in order of *priority*:
  1. ***Continue despite loss of networks or gateways***
  2. Support multiple types of communication service
  3. Accommodate a variety of networks
  4. Permit distributed management of Internet resources
  5. Cost effective
  6. Host attachment should be easy
  7. Resource accountability

Survivability

- If network disrupted and reconfigured
  - Communicating entities should not care!
  - This means:
    - Transport interface only knows "working" and "not working"
    - Not working == complete partition.
    - Mask all transient failures
- How to achieve such reliability?
  - State info for on-going conversation must be protected
  - Where can communication state be stored?
    - If lower layers lose it → app gets affected
    - Store at lower layers and replicate
      - But effective replication is hard

Fate Sharing



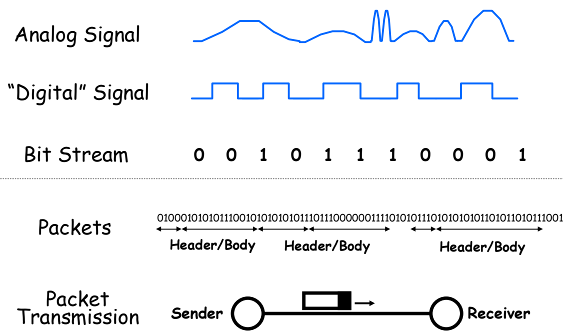
- Lose state information for an entity if (and only if?) the entity itself is lost
  - Protects from intermediate failures
  - Easier to engineer than replication
  - Switches are stateless
- Examples:
  - OK to lose TCP state if one endpoint crashes
    - NOT okay to lose if an intermediate router reboots

CS 640: Introduction to Computer Networks

Aditya Akella

Lecture 5 -
Encoding and Data Link Basics

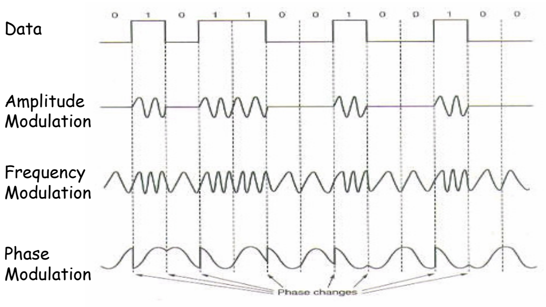
Signals, Data and Packets



Binary data to Signals

- Modulation: changing attributes of signal to effect information transmissions
- Encoding
  - How to convert bits to "digital" signals
  - Very complex, actually
  - Error recovery, clock recovery,…

Modulation Schemes



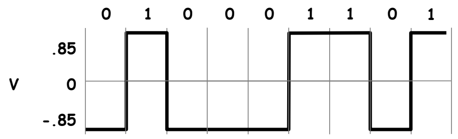
Why Do We Need Encoding?

- Meet certain electrical constraints.
  - Receiver needs enough "transitions" to keep track of the transmit clock
  - Avoid receiver saturation
- Create control symbols, besides regular data symbols.
  - E.g. start or end of frame, escape, ...
  - Important in packet switching
- 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 very complex, e.g. wireless.

Encoding

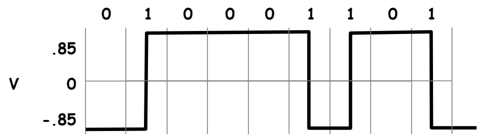
- Use two signals, high and low, to encode 0 and 1.
- Transmission is synchronous, i.e., a clock is used to sample the signal.
  - In general, the duration of one bit is equal to one or two clock ticks
  - Receiver's clock must be synchronized with the sender's clock
- Encoding can be done one bit at a time or in blocks of, e.g., 4 or 8 bits.

Non-Return to Zero (NRZ)



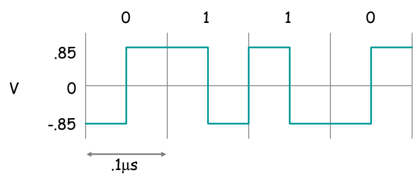
- 1 -> high signal; 0 -> low signal
- Long sequences of 1's or 0's can cause problems:
  - Hard to recover clock
  - Difficult to interpret 0's and 1's

Non-Return to Zero Inverted (NRZI)



- 1 -> make transition; 0 -> signal stays the same
- Solves the problem for long sequences of 1's, but not for 0's.

Ethernet Manchester Encoding



- Positive transition for 0, negative for 1
- XOR of NRZ with clock
- Transition every cycle communicates clock (but need 2 transition times per bit)
- Problem: doubles the rate at which signal transitions are made
  - Less efficient
  - Receiver has half the time to detect the pulse

4B/5B Encoding

- Data coded as *symbols* of 5 line bits => 4 data bits, so 100 Mbps uses 125 MHz.
  - Uses less frequency space than Manchester encoding
- Each valid symbol has no more than one leading zero and no more than two trailing zeros
  - At least two 1s ➔ Get dense transitions
- Uses NRZI to encode the 5 code bits
  - What happens if there are consecutive 1s?
- Example: FDDI.

4B/5B Encoding

- 16 data symbols, 8 control symbols
  - Control symbols: idle, begin frame, etc.
  - Remaining 8 are invalid

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

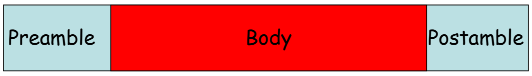
Other Encodings

- 8B/10B: Fiber Channel and Gigabit Ethernet
  - DC balance
- 64B/66B: 10 Gbit Ethernet
- B8ZS: T1 signaling (bit stuffing)

Datalink Protocol Functions

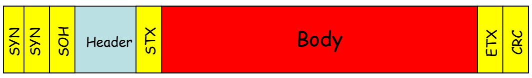
1. **Framing**: encapsulating a network layer
   - Add header, mark and detect frame boundaries, …
2. **Error control**: error detection and correction to deal with bit errors.
   - May also include other reliability support, e.g. retransmission
3. **Error correction**: Correct bit errors if possible
4. **Flow control**: avoid sender outrunning the receiver.
5. **Media access**: controlling which frame should be sent over the link next
   - Easy for point-to-point links
     - Half versus full duplex
   - Harder for multi-access links
     - Who gets to send?
6. **Switching**: How to send frames to the eventual destination?

Framing



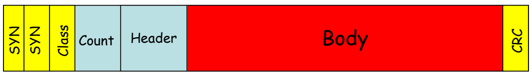
- A link layer function, defining which bits have which function
- *Minimal functionality*: mark the beginning and end of packets (or frames).
- Some techniques:
  - frame delimiter characters with character stuffing
  - frame delimiter codes with bit stuffing
  - synchronous transmission (e.g. SONET) out of band delimiters

Byte Stuffing



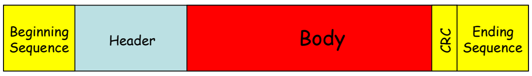
- Mark end of frame with special character
  - BISYNC uses "ETX"
  - What happens when the user sends this character?
    - Use escape character when controls appear in data
  - Very common on serial lines; old technique
  - View frame as a collection of bytes

Byte Counting



- An alternative is to include a count of number of bytes
  - Next to the start of frame
  - E.g. DDCMP
  - Corruptions of count field may cause receiver to receive incorrectly
  - Include an error-check to help receiver realize this

Bit Stuffing



- Treat frames as a sequence of bits
- Mark frames with special bit sequence
  - Example, HDLC: 01111110 is a special sequence or "flag"
    - Used at the beginning and end of frame
  - But, must ensure data containing this sequence can be transmitted
    - Flag can cross byte boundaries
  - transmitter inserts a 0 when this is likely to appear in the data:
    - 111111 -> 1111101
    - must stuff a zero any time five 1s appear:
  - receiver unstuffs.
- Problem with stuffing techniques: frame size depends on data
  - Frames can be of different size
  - Could lead to some inefficiencies

CS 640: Computer Networks

Aditya Akella

Lecture 6 -
Error/Flow Control
&
Intro to Switching
and Medium Access Control

Error Coding

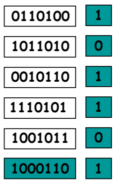
- Transmission process may introduce errors into a message.
  - Single bit errors versus burst errors
- Detection: e.g. CRC
  - Requires a check that some messages are invalid
  - Hence requires extra bits
  - "redundant check bits"
- Correction
  - Forward error correction: many related code words map to the same data word
  - Detect errors and retry transmission

Parity

- Even parity
  - Append parity bit to 7 bits of data to make an even number of 1's
  - Odd parity accordingly defined. 1010100 | 1 ; 1001011 | 0
- 1 in 8 bits of overhead?
  - When is this a problem?
- Can detect a single error 1010101 | 1
- But nothing beyond that 1000010 | 0

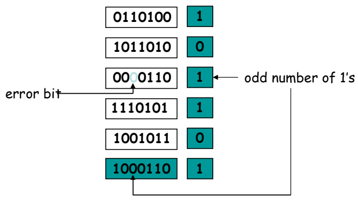
2-D Parity

- Make each byte even parity
- Finally, a parity byte for all bytes of the packet
- Example: five 7-bit character packet, even parity



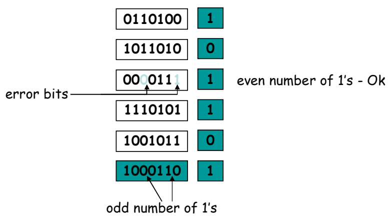
Effectiveness of 2-D Parity

- 1-bit errors can be detected, corrected
- Example with even parity per byte:



Effectiveness of 2-D Parity

- 2-bit errors can also be detected
- Example:



- What about 3-bit errors? >3-bit errors?

Cyclic Redundancy Codes (CRC)

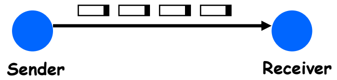
- Commonly used codes that have good error detection properties
  - Can catch many error combinations with a small number or redundant bits
- Based on division of polynomials
  - Errors can be viewed as adding terms to the polynomial
  - Should be unlikely that the division will still work
- Can be implemented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

Link Flow Control and Error Control

- Dealing with receiver overflow: flow control.
- Dealing with packet loss and corruption: error control.
- Actually these issues are relevant at many layers.
  - Link layer: sender and receiver attached to the same "wire"
  - End-to-end: transmission control protocol (TCP) - sender and receiver are the end points of a connection
- How can we implement flow control?
  - "You may send" (windows, stop-and-wait, etc.)
  - "Please shut up" (source quench, 802.3x pause frames, etc.)

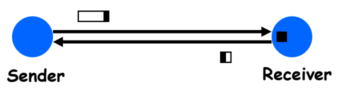
Flow Control: A Naïve Protocol

- Sender simply sends to the receiver whenever it has packets.
- Potential problem: sender can outrun the receiver.
  - Receiver too slow, small buffer overflow, ..
- Not always a problem: receiver might be fast enough.



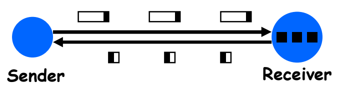
Adding Flow Control

- Stop and wait flow control: sender waits to send the next packet until the previous packet has been acknowledged by the receiver.
  - Receiver can pace the sender
- Drawbacks: adds overheads, slowdown for long links.



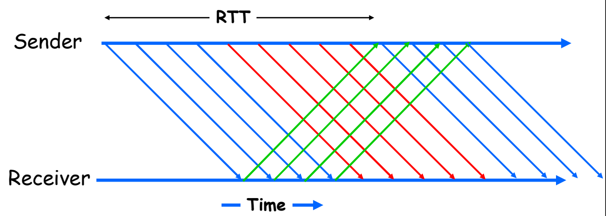
Window Flow Control

- Stop and wait flow control results in poor throughput for long-delay paths: packet size/ roundtrip-time.
- Solution: receiver provides sender with a window that it can fill with packets.
  - The window is backed up by buffer space on receiver
  - Receiver acknowledges the a packet every time a packet is consumed and a buffer is freed



Window Limitations

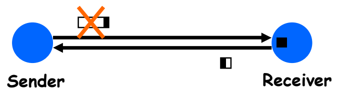
Window Size = 4pkts



$$\text{Throughput} = \frac{\text{Window Size}}{\text{Roundtrip Time}}$$

Error Control: Stop and Wait Case

- Packets can get lost, corrupted, or duplicated.
- Duplicate packet: use sequence numbers.
- Lost packet: time outs and acknowledgements.
  - Positive versus negative acknowledgements
  - Sender side versus receiver side timeouts
- Window based flow control: more aggressive use of sequence numbers (see transport lectures).



What is Used in Practice?

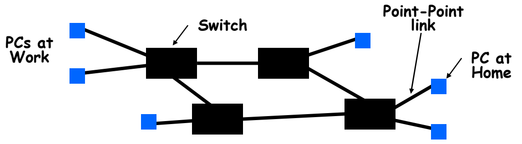
- No flow or error control.
  - E.g. regular Ethernet, just uses CRC for error detection
- Flow control only.
  - E.g. Gigabit Ethernet
- Flow and error control.
  - E.g. X.25 (older connection-based service at 64 Kbs that guarantees reliable in order delivery of data)

Switching and Media Access Control

- How do we transfer packets between two hosts connected to the a switched network?
- Switches connected by point-to-point links -- store-and-forward.
  - Multiplexing and forwarding
  - Used in WAN, LAN, and for home connections
  - Conceptually similar to "routing"
    - But at the datalink layer instead of the network layer
- Multiple access networks -- contention based.
  - Multiple hosts are sharing the same transmission medium
  - Used in LANs and wireless
  - Need to control access to the medium

A Switch-based Network

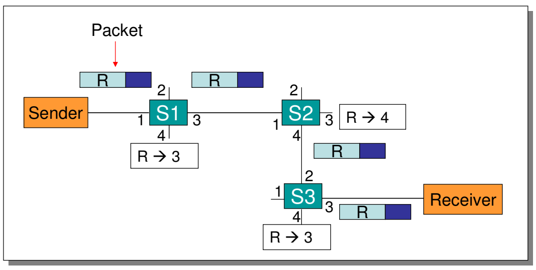
- Switches are connected by "point-to-point" links.
- Packets are forwarded hop-by-hop by the switches towards the destination.
  - Many forms of forwarding
- Many datalink technologies use switching.
  - Virtual circuits: Frame-relay, ATM, X.25, ..
  - Packets: Ethernet, MPLS, …



Three techniques for switching

- Global addresses - connection-less
  - Routers keep next hop for destination
  - Packets carry destination address
- Virtual circuits – connection oriented
  - Connection routed through network to set up state
  - Packets forwarded using connection state
- Source routing
  - Packet carries path

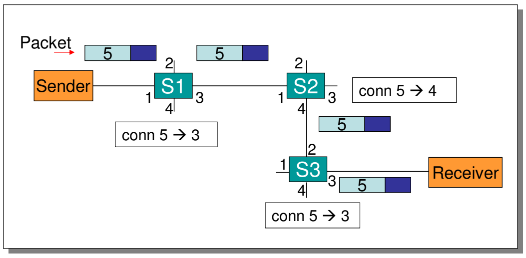
Global Address Example



Global Addresses

- Advantages
  - Stateless – simple error recovery
- Disadvantages
  - Every switch knows about every destination
    - Potentially large tables
  - All packets to destination take same route
  - Need special approach to fill table

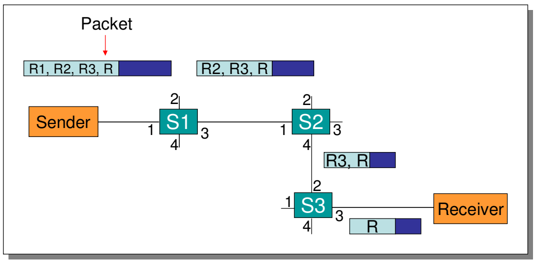
Simplified Virtual Circuits Example



Virtual Circuits

- Advantages
  - Efficient lookup (simple table lookup)
  - Can reserve bandwidth at connection setup
  - Easier for hardware implementations
- Disadvantages
  - Still need to route connection setup request
  - More complex failure recovery – must recreate connection state
- Typical use → fast router implementations
  - ATM – combined with fix sized cells
  - MPLS – tag switching for IP networks

Source Routing Example



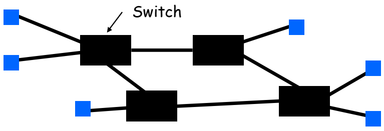
Source Routing

- Advantages
  - Switches can be very simple and fast
- Disadvantages
  - Variable (unbounded) header size
  - Sources must know or discover topology (e.g., failures)
- Typical uses
  - Ad-hoc networks (DSR)
  - Machine room networks (Myrinet)

Comparison

	Source Routing	Global Addresses	Virtual Circuits
Header Size	Worst	OK – Large address	Best
Router Table Size	None	Number of hosts	Number of circuits
Forward Overhead	Best	Table lookup	Pretty Good
Setup Overhead	None	None	Connection Setup
Error Recovery	Tell all hosts	Tell all switches	Tell all switches and Tear down circuit and re-route

Most Popular: Address Lookup-based Approach



Address	Next Hop	Info
B31123812508	3	13
38913C3C2137	3	-
A21023C90590	0	-
128.2.15.3	1	(2,34)

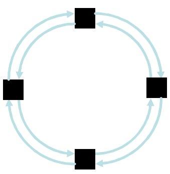
- Address from header.
  - Absolute address (e.g. Ethernet)
  - (IP address for routers)
  - (VC identifier, e.g. ATM))
- Next hop: output port for packet.
- Info: priority
- We will see how this table is filled (learning bridges)

Multiple Access Protocols

- Prevent two or more nodes from transmitting at the same time over a broadcast channel.
  - If they do, we have a collision, and receivers will not be able to interpret the signal
- Several classes of multiple access protocols.
  - Partitioning the channel, e.g. frequency-division or time division multiplexing
    - With fixed partitioning of bandwidth – not flexible
  - Taking turns, e.g. token-based, reservation-based protocols, polling based
  - Contention based protocols, e.g. Aloha, Ethernet
    - Next lecture

Fiber Distributed Data Interface (FDDI)

- One token holder may send, with a time limit.
  - known upper bound on delay.
- Optical version of 802.5 token ring, but multiple packets may travel in train: token released at end of frame.
- 100 Mbps, 100km.
- Optional dual ring for fault tolerance.
- CDDI: FDDI over unshielded twisted pair, shorter range



Other "Taking Turn" Protocols

- Central entity polls stations, inviting them to transmit.
  - Simple design – no conflicts
  - Not very efficient – overhead of polling operation
- Stations reserve a slot for transmission.
  - For example, break up the transmission time in contention-based and reservation based slots
    - Contention based slots can be used for short messages or to reserve time
    - Communication in reservation based slots only allowed after a reservation is made
  - Issues: fairness, efficiency

CS 640: Introduction to Computer Networks

Aditya Akella

Lecture 7 -
Ethernet, Bridges,
Learning and Spanning Tree

Multiple Access Protocols

- Prevent two or more nodes from transmitting at the same time over a *broadcast* channel.
  - If they do, we have a *collision*, and receivers will not be able to interpret the signal
- Several classes of multiple access protocols.
  - Partitioning the channel, e.g. frequency-division or time division multiplexing
  - Taking turns, e.g. token-based, reservation-based protocols, polling based
  - Contention based protocols, e.g. Aloha, Ethernet

Desirable MAC Properties

Broadcast channel of capacity R bps.

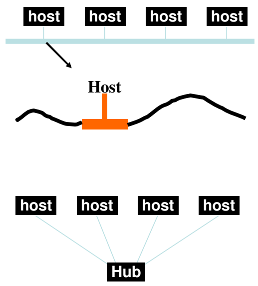
- 1 node ➔ throughput = R bps
- N nodes ➔ throughput = R/N bps, on average
- Decentralized
- Simple, inexpensive

Contention-Based Protocols

- Idea: access the channel in a "random" fashion - when collisions occur, recover.
  - Each node transmits at highest rate of R bps
  - Collision: two or more nodes transmitting at the same time
    - Each node retransmits until collided packet gets through
  - Key: don't retransmit right away
    - Wait a random interval of time first
- Examples
  - Aloha
  - Ethernet – focus today

Ethernet Physical Layer

- 10Base2 standard based on thin coax → 200m
  - Nodes are connected using thin coax cables and BNC "T" connectors in a bus topology
  - Thick coax no longer used
- 10BaseT uses twisted pair and hubs → 100m
  - Stations can be connected to each other or to hubs
  - Hub acts as a concentrator
    - Dumb device
- The two designs have the same protocol properties.
  - Key: electrical connectivity between all nodes
  - Deployment is different



Ethernet Frame Format

8	6	6	2			4
Preamble	Dest	Source	Type	Data	Pad	CRC

- Preamble marks the beginning of the frame.
  - Also provides synchronization
- Source and destination are 48 bit IEEE MAC addresses.
  - Flat address space
  - Hardwired into the network interface
- Type field is a demultiplexing field.
  - What network layer (layer 3) should receive this packet?
- Max frame size = 1500B; min = 46B
  - Need padding to meet min requirement
- CRC for error checking.

Ethernet host side

- Transceiver: detects when the medium is idle and transmits the signal when host wants to send
  - Connected to "Ethernet adaptor"
  - Sits on the host
- Any host signal broadcast to everybody
  - But transceiver accepts frames addressed to itself
  - Also frames sent to broadcast address
  - All frames, if in promiscuous mode
- When transmitting, all hosts on the same segment, or connected to the same hub, compete for medium
  - Said to "share same collision domain"
  - Bad for efficiency!

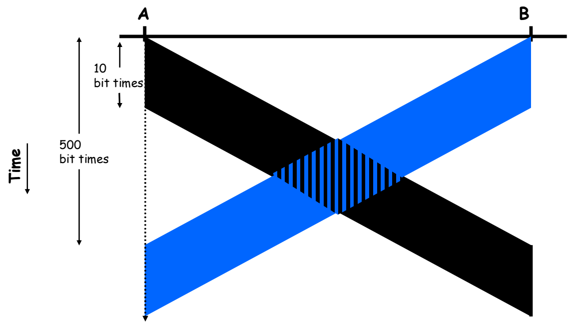
Sender-side: MAC Protocol

- Carrier-sense multiple access with collision detection (CSMA/CD).
  - MA = multiple access
  - CS = carrier sense
  - CD = collision detection

CSMA/CD Algorithm Overview

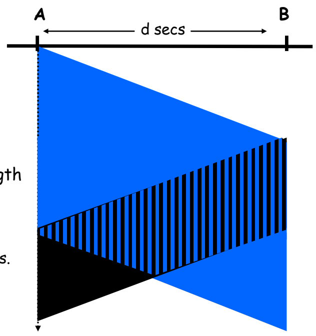
- Sense for carrier.
  - "Medium idle"?
- If medium busy, wait until idle.
  - Sending would force a collision and waste time
- Send packet and sense for collision.
- If no collision detected, consider packet delivered.
- Otherwise, abort immediately, perform *exponential back off* and send packet again.
  - Start to send after a random time picked from an interval
  - Length of the interval increases with every collision, retransmission attempt

Collision Detection



Collision Detection: Implications

- All nodes must be able to detect the collision.
  - Any node can be sender
- => Must either have short wires, long packets, or both
- If A starts at t, and wirelength is d secs,
  - In the worst case, A may detect collision at t+2d
  - ➔ Will have to send for 2d secs.
  - ➔ d depends on max length of ethernet cable



Minimum Packet Size

- Give a host enough time to detect a collision.
- In Ethernet, the minimum packet size is 64 bytes.
  - 18 bytes of header and 46 data bytes
  - If the host has less than 46 bytes to send, the adaptor pads bytes to increase the length to 46 bytes
- What is the relationship between the minimum packet size and the size of LAN?

  *LAN size = (min frame size) * light speed / (2 * bandwidth)*
- How did they pick the minimum packet size?

CSMA/CD: Some Details

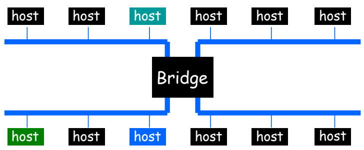
- When a sender detects a collision, it sends a "jam signal".
  - Make sure that all nodes are aware of the collision
  - Length of the jam signal is 32 bit times
  - Permits early abort - don't waste max transmission time
- Exponential backoff operates in multiples of 512 bit times.
  - RTT= 256bit times ➔ backoff time > Longer than a roundtrip time
  - Guarantees that nodes that back off will notice the earlier retransmission before starting to send
- Successive frames are separated by an "inter-frame" gap.
  - to allow devices to prepare for reception of the next frame
  - Set to 9.6 µsec or 96 bit times

LAN Properties

- Exploit physical proximity.
  - Often a limitation on the physical distance
  - E.g. to detect collisions in a contention based network
- Relies on single administrative control and some level of trust.
  - Broadcasting packets to everybody and hoping everybody (other than the receiver) will ignore the packet
- Broadcast: nodes can send messages that can be heard by all nodes on the network.
  - Almost essential for network administration
  - Can also be used for applications, e.g. video conferencing
- But broadcast fundamentally does not scale.

Building Larger LANs: Bridges

- Hubs are physical level devices
  - Don't isolate collision domains ➔ broadcast issues
- At layer 2, *bridges* connect multiple IEEE 802 LANs
  - Separate a single LAN into multiple smaller collision domains
    - Reduce collision domain size



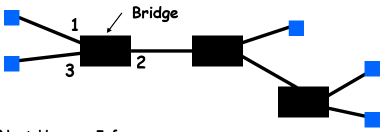
Basic Bridge Functionality

- Bridges are full fledged packet switches
- Frame comes in on an interface
  - Switch looks at destination LAN address
  - Determines port on which host connected
  - Only forward packets to the right port
  - Must run CSMA/CD with hosts connected to same LAN
    - Also between bridge and host connected to a LAN

"Transparent" Bridges

- Design features:
  - "Plug and play" capability
  - Self-configuring without hardware or software changes
  - Bridge do not impact the operation of the individual LANs
- Three components of transparent bridges:
  1) Forwarding of frames
  2) Learning of addresses
  3) Spanning tree algorithm

Address Lookup/Forwarding Example

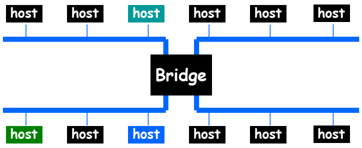


Address	Next Hop	Info
A21032C9A591	1	8:36
99A323C90842	2	8:01
8711C98900AA	2	8:15
301B2369011C	2	8:16
695519001190	3	8:11

- Address is a 48 bit IEEE MAC address.
- Next hop: output port for packet
- Timer is used to flush old entries
- Size of the table is equal to the number of hosts
- Flat address → no aggregation
- No entry ➔ packets are broadcasted

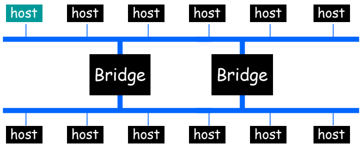
Learning

- Bridge tables can be filled in manually (flush out old entries etc)
  - Time consuming, error-prone
  - Self-configuring preferred
    - Bridges use "*learning*"
- Keep track of source address of packet (S) and the arriving interface (I).
  - Fill in the forwarding table based on this information
  - Packet with destination address S must be sent to interface I!



Spanning Tree Bridges

- More complex topologies can provide redundancy.
  - But can also create loops.
    - E.g. What happens when there is no table entry?
  - Multiple copies of data
  - → Could crash the network → has happened often!



Spanning Tree Protocol Overview

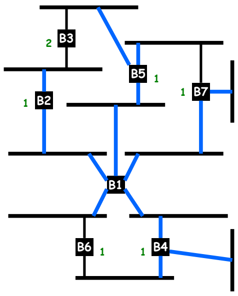
Embed a tree that provides a single unique default path to each destination:

Bridges designate ports over which they will or will not forward frames

By removing ports, extended LAN is reduced to a tree

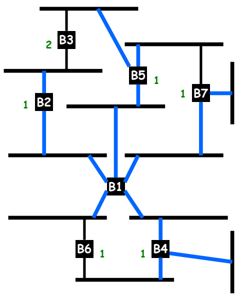
Spanning Tree Algorithm

- Root of the spanning tree is elected first → the bridge with the lowest identifier.
  - All ports are part of tree
- Each bridge finds shortest path to the root.
  - Remembers port that is on the shortest path
  - Used to forward packets
- Select for each LAN a designated bridge that will forward frames to root
  - Has the shortest path to the root.
  - Identifier as tie-breaker



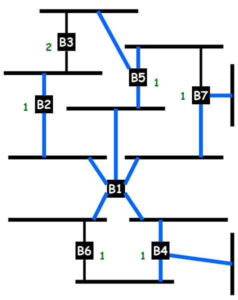
Spanning Tree Algorithm

- Each node sends configuration message to all neighbors.
  - Identifier of the sender
  - Id of the presumed root
  - Distance to the presumed root
- Initially each bridge thinks it is the root.
  - B5 sends (B5, B5, 0)
- When B receive a message, it decide whether the solution is better than their local solution.
  - A root with a lower identifier?
  - Same root but lower distance?
  - Same root, distance but sender has lower identifier?
- Message from bridge with smaller root ID
  - Not root; stop generating config messages, but can forward
- Message from bridge closer to root
  - Not designated bridge; stop sending any config messages on the port



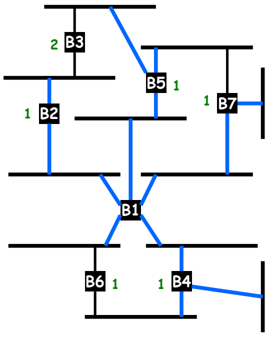
Spanning Tree Algorithm

- Each bridge B can now select which of its ports make up the spanning tree:
  - B's root port
  - All ports for which B is the designated bridge on the LAN
- States for ports on bridges
  - *Forward state* or *blocked state*, depending on whether the port is part of the spanning tree
- Root periodically sends configuration messages and bridges forward them over LANs they are responsible for



Spanning Tree Algorithm Example

- Node B2:
  - Sends (B2, B2, 0)
  - Receives (B1, B1, 0) from B1
  - Sends (B2, B1, 1) "up"
  - Continues the forwarding forever
- Node B1:
  - Will send notifications forever
- Node B7:
  - Sends (B7, B7, 0)
  - Receives (B1, B1, 0) from B1
  - Sends (B7, B1, 1) "up" and "right"
  - Receives (B5, B5, 0) - ignored
  - Receives (B5, B1, 1) – suboptimal
  - Continues forwarding the B1 messages forever to the "right"



Ethernet Switches

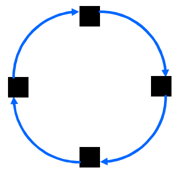
- Bridges make it possible to increase LAN capacity.
  - Packets are no longer broadcasted - they are only forwarded on selected links
  - Adds a switching flavor to the broadcast LAN
  - Some packets still sent to entire tree (e.g., ARP)
- Ethernet switch is a special case of a bridge: each bridge port is connected to a single host.
  - Can make the link full duplex (really simple protocol!)
  - Simplifies the protocol and hardware used (only two stations on the link) – no longer full CSMA/CD
  - Can have different port speeds on the same switch
    - Unlike in a hub, packets can be stored

A Word about "Taking Turn" Protocols

- First option: Polling-based
  - Central entity polls stations, inviting them to transmit.
    - Simple design – no conflicts
    - Not very efficient – overhead of polling operation
    - Still better than TDM or FDM
    - Central point of failure
- Second (similar) option: Stations reserve a slot for transmission.
  - For example, break up the transmission time in contention-based and reservation based slots
    - Contention based slots can be used for short messages or to reserve time
    - Communication in reservation based slots only allowed after a reservation is made
  - Issues: fairness, efficiency

Token-Passing Protocols

- No master node
  - Fiber Distributed Data Interface (FDDI)
- One token holder may send, with a time limit.
  - known upper bound on delay.
- Token released at end of frame.
  - 100 Mbps, 100km
- Decentralized and very efficient
  - But problems with token holding node crashing or not releasing token



CS 640: Introduction to Computer Networks

Aditya Akella

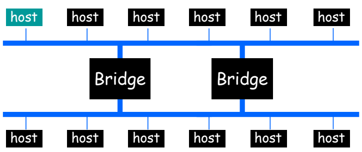
Lecture 7 -
IP: Addressing and Forwarding

From the previous lecture…

- We will cover spanning tree from the last lecture

Spanning Tree Bridges

- More complex topologies can provide redundancy.
  - But can also create loops.
    - E.g. What happens when there is no table entry?
  - Multiple copies of data
  - → Could crash the network → has happened often!



Spanning Tree Protocol Overview

Embed a tree that provides a single unique default path to each destination:

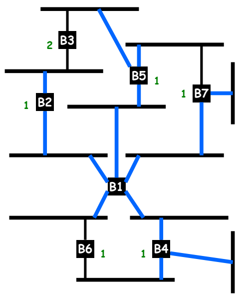
Bridges designate ports over which they will or will not forward frames

By removing ports, extended LAN is reduced to a tree

4

Spanning Tree Algorithm

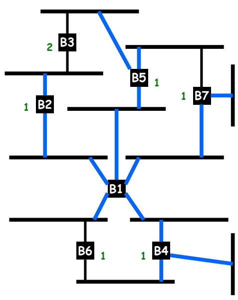
- Root of the spanning tree is elected first → the bridge with the lowest identifier.
  - All ports are part of tree
- Each bridge finds shortest path to the root.
  - Remembers port that is on the shortest path
  - Used to forward packets
- Select for each LAN a designated bridge that will forward frames to root
  - Has the shortest path to the root.
  - Identifier as tie-breaker



5

Spanning Tree Algorithm

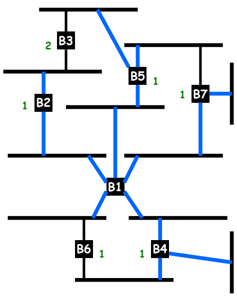
- Each node sends configuration message to all neighbors.
  - Identifier of the sender
  - Id of the presumed root
  - Distance to the presumed root
- Initially each bridge thinks it is the root.
  - B5 sends (B5, B5, 0)
- When B receive a message, it decide whether the solution is better than their local solution.
  - A root with a lower identifier?
  - Same root but lower distance?
  - Same root, distance but sender has lower identifier?
- Message from bridge with smaller root ID
  - Not root; stop generating config messages, but can forward
- Message from bridge closer to root
  - Not designated bridge; stop sending any config messages on the port



6

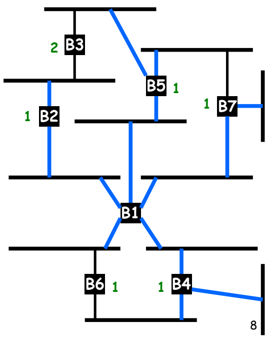
Spanning Tree Algorithm

- Each bridge B can now select which of its ports make up the spanning tree:
  - B's root port
  - All ports for which B is the designated bridge on the LAN
- States for ports on bridges
  - *Forward state* or *blocked state*, depending on whether the port is part of the spanning tree
- Root periodically sends configuration messages and bridges forward them over LANs they are responsible for



Spanning Tree Algorithm Example

- Node B2:
  - Sends (B2, B2, 0)
  - Receives (B1, B1, 0) from B1
  - Sends (B2, B1, 1) "up"
  - Continues the forwarding forever
- Node B1:
  - Will send notifications forever
- Node B7:
  - Sends (B7, B7, 0)
  - Receives (B1, B1, 0) from B1
  - Sends (B7, B1, 1) "up" and "right"
  - Receives (B5, B5, 0) - ignored
  - Receives (B5, B1, 1) – suboptimal
  - Continues forwarding the B1 messages forever to the "right"



Ethernet Switches

- Bridges make it possible to increase LAN capacity.
  - Packets are no longer broadcasted - they are only forwarded on selected links
  - Adds a switching flavor to the broadcast LAN
  - Some packets still sent to entire tree (e.g., ARP)
- Ethernet switch is a special case of a bridge: each bridge port is connected to a single host.
  - Can make the link full duplex (really simple protocol!)
  - Simplifies the protocol and hardware used (only two stations on the link) – no longer full CSMA/CD
  - Can have different port speeds on the same switch
    - Unlike in a hub, packets can be stored

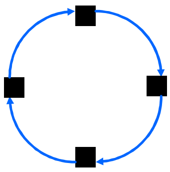
A Word about "Taking Turn" Protocols

- First option: Polling-based
  - Central entity polls stations, inviting them to transmit.
    - Simple design – no conflicts
    - Not very efficient – overhead of polling operation
    - Still better than TDM or FDM
    - Central point of failure
- Second (similar) option: Stations reserve a slot for transmission.
  - For example, break up the transmission time in contention-based and reservation based slots
    - Contention based slots can be used for short messages or to reserve time
    - Communication in reservation based slots only allowed after a reservation is made
  - Issues: fairness, efficiency

10

Token-Passing Protocols

- No master node
  - Fiber Distributed Data Interface (FDDI)
- One token holder may send, with a time limit.
  - known upper bound on delay.
- Token released at end of frame.
  - 100 Mbps, 100km
- Decentralized and very efficient
  - But problems with token holding node crashing or not releasing token



11

This Lecture: IP addressing and Forwarding

12

Simple Internetworking

- Focus on a single internetwork
  - Internetwork = combo of multiple physical networks
- How do I designate hosts?
  - Addressing
- How do I send information to a distant host?
  - Underlying service model
    - What gets sent?
    - How fast will it go? What happens if it doesn't get there?
  - Routing/Forwarding
    - **Global addresses-based** forwarding is used
    - What path is it sent on?
    - How is this path computed?

13

Addressing in IP: Considerations

- Uniquely designate hosts
  - MAC addresses may do, but they are useless for scalable routing
- Hierarchical vs. flat
  - Wisconsin / Madison / UW-Campus / Aditya
    vs.
    Aditya:123-45-6789
  - Ethernet addresses are flat
  - IP addresses are hierarchical
- Why Hierarchy?
  - Scalable routing
  - Route to a general area, then to a specific location

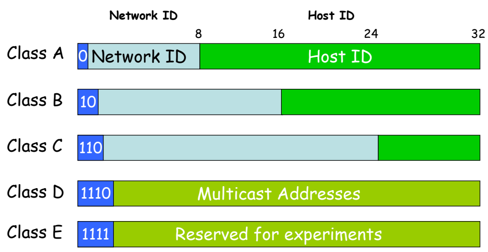
14

IP Addresses

- Fixed length: 32 bits
- Total IP address size: 4 billion
- Initial class-ful structure (1981)
  - Class A: 128 networks, 16M hosts
  - Class B: 16K networks, 64K hosts
  - Class C: 2M networks, 256 hosts

15

IP Address Classes
(Some are Obsolete)



16

Original IP Route Lookup

- Address would specify prefix for forwarding table
  - Simple lookup
- www.cmu.edu address 128.2.11.43
  - Class B address – class + network is 128.2
  - Lookup 128.2 in forwarding table
  - Prefix – part of address that really matters for routing
- Forwarding table contains
  - List of class+network entries
  - A few fixed prefix lengths (8/16/24)
- Large tables
  - 2 Million class C networks

17

Example

- Host: Get n/w number for destination: Nd → compare with sending host n/w number

N/W number	Outgoing Interface
N	Eth0
Default	R1

- Router: Compare dest n/w number with n/w number of each interface → either put on interface, or send to next hop router

N/W number	Outgoing Interface
N0	Eth0
N1	Eth1
N2	R2
N3	R3

18

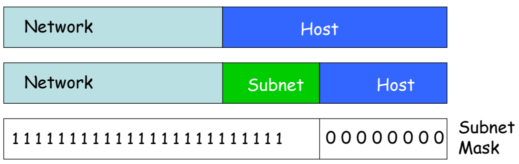
Subnet Addressing: RFC917 (1984)

- Original goal: network part would uniquely identify a single physical network
- Inefficient address space usage
  - Class A & B networks too big
    - Also, very few LANs have close to 64K hosts
    - Easy for networks to (claim to) outgrow class-C
  - Each physical network must have one network number
- Routing table size is too high
- Need simple way to reduce the number of network numbers assigned
  - Subnetting: Split up single network address ranges
  - Fizes routing table size problem, partially

19

Subnetting

- Add another "floating" layer to hierarchy
- Variable length subnet masks
  - Could subnet a class B into several chunks



20

Subnetting Example

- Assume an organization was assigned address 150.100 (class B)
- Assume < 100 hosts per subnet (department)
- How many host bits do we need?
  - Seven
- What is the network mask?
  - 11111111 11111111 11111111 10000000
  - 255.255.255.128

21

Forwarding Example

- Host configured with IP adress and subnet mask
- Subnet number = IP (AND) Mask
- (Subnet number, subnet mask) → Outgoing I/F

```
D = destination IP address
For each forwarding table entry (SN, SM → OI)
  D1 = SM & D
  if (D1 == SN)
    if nexthop is interface
        Deliver on INTERFACE
    Else
        Forward to default router
```

22

Inefficient Address Usage

- Address space depletion
  - In danger of running out of classes A and B
  - Why?
    - Class C too small for most domains
    - Very few class A – very careful about giving them out
    - Class B poses greatest problem
  - Class B sparsely populated
    - But people refuse to give it back

23

Classless Inter-Domain Routing (CIDR) – RFC1338

- Allows arbitrary split between network & host part of address
  - Do not use classes to determine network ID
  - Use common part of address as network number
  - Allows handing out arbitrary sized chunks of address space
  - E.g., addresses 192.4.16 - 192.4.31 have the first 20 bits in common. Thus, we use these 20 bits as the network number → 192.4.16/20
- Enables more efficient usage of address space (and router tables)
  - Use single entry for range in forwarding tables
  - Combine forwarding entries when possible

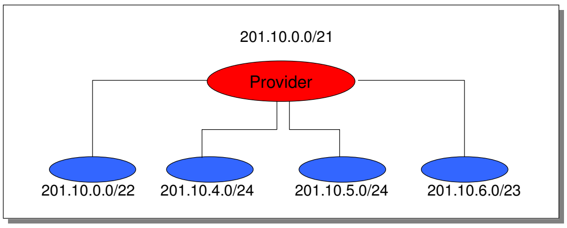
24

CIDR Example

- Network is allocated 8 contiguous chunks of 256-host addresses 200.10.0.0 to 200.10.7.255
  - Allocation uses 3 bits of class C space
  - Remaining 21 bits are network number, written as 201.10.0.0/21
- Replaces 8 class C routing entries with 1 combined entry
  - Routing protocols carry prefix with destination network address

25

CIDR Illustration



26

CIDR Implications

- Longest prefix match
  - 7 contiguous Class C's given to network A:
    - 200.10.0.0 – 200.10.6.255
    - N/w number – 200.10.0.0/21
  - 8th class C given to network B:
    - 200.10.7.0 – 200.10.7.255
    - N/w number – 200.10.7.0/24
  - Packet with destination address 200.10.7.1 matches both networks
    - Must pick the most specific match!

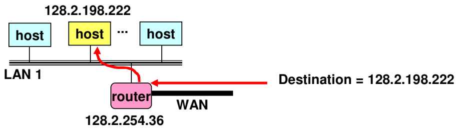
27

CS 640: Introduction to Computer Networks

Aditya Akella

Lecture 9 -
ARP, IP Packets and Routers

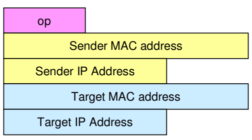
Finding a Local Machine



- Routing Gets Packet to Correct Local Network
  - Based on IP address
  - Router sees that destination address is of local machine
- Still Need to Get Packet to Host
  - Using link-layer protocol
  - Need to know hardware address
- Same Issue for Any Local Communication
  - Find local machine, given its IP address

2

Address Resolution Protocol (ARP)

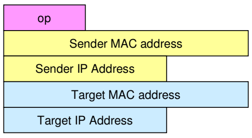


- op: Operation
  - 1: request
  - 2: reply
- Sender
  - Host sending ARP message
- Target
  - Intended receiver of message

  - Diagrammed for Ethernet (6-byte MAC addresses)
- Low-Level Protocol
  - Operates only within local network
  - Determines mapping from IP address to hardware (MAC) address
  - Mapping determined dynamically
    - No need to statically configure tables
    - Only requirement is that each host know its own IP address

3

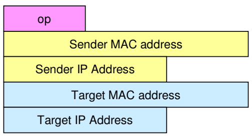
ARP Request



- op: Operation
  - 1: request
- Sender
  - Host that wants to determine MAC address of another machine
- Target
  - Other machine

- Requestor
  - Fills in own IP and MAC address as "sender"
    - Why include its MAC address?
- Mapping
  - Fills desired host IP address in target IP address
- Sending
  - Send to MAC address `ff:ff:ff:ff:ff:ff`
    - Ethernet broadcast

ARP Reply



- op: Operation
  - 2: reply
- Sender
  - Host with desired IP address
- Target
  - Original requestor

- Responder becomes "sender"
  - Fill in own IP and MAC address
  - Set requestor as target
  - Send to requestor's MAC address

IP Delivery Model

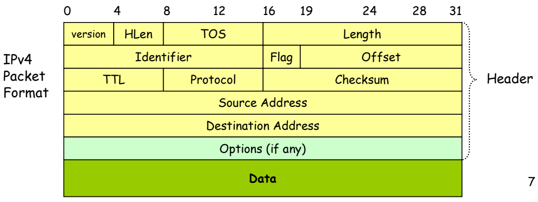
- *Best effort service*
  - Network will do its best to get packet to destination

- Does NOT guarantee:
  - Any maximum latency or even ultimate success
  - Sender will be informed if packet doesn't make it
  - Packets will arrive in same order sent
  - Just one copy of packet will arrive

- Implications
  - Scales very well → simple, dumb network; "plug-n-play"
  - Higher level protocols must make up for shortcomings
    - Reliably delivering ordered sequence of bytes → TCP
  - Some services not feasible
    - Latency or bandwidth guarantees
    - Need special support

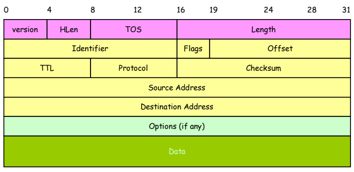
IP Packets

- Low-level communication model provided by Internet
  - Unit: “Datagram”
- Datagram
  - Each packet self-contained
    - All information needed to get to destination
  - Analogous to letter or telegram



7

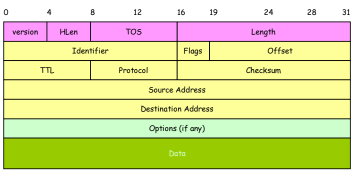
IPv4 Header Fields



- Version: IP Version
  - 4 for IPv4
  - 6 for IPv6
- HLen: Header Length
  - 32-bit words (typically 5)
- TOS: Type of Service
  - Priority information
- Length: Packet Length
  - Bytes (including header)
- Header format can change with versions
  - First byte identifies version
  - IPv6 header are very different – will see later
- Length field limits packets to 65,535 bytes
  - In practice, break into much smaller packets for network performance considerations

8

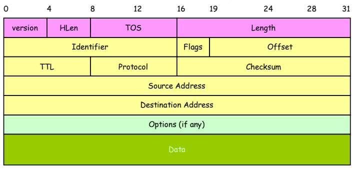
IPv4 Header Fields



- Identifier, flags, fragment offset → used primarily for fragmentation
- Time to live
  - Must be decremented at each router
  - Packets with TTL=0 are thrown away
  - Ensure packets exit the network
- Protocol
  - Demultiplexing to higher layer protocols
  - TCP = 6, ICMP = 1, UDP = 17…
- Header checksum
  - Ensures some degree of header integrity
  - Relatively weak – only 16 bits
- Options
  - E.g. Source routing, record route, etc.
  - Performance issues at routers
    - Poorly supported or not at all

9

IPv4 Header Fields

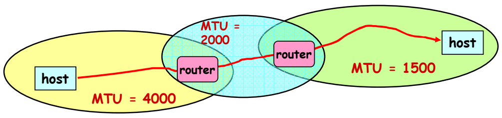


- Source Address
  - 32-bit IP address of sender
- Destination Address
  - 32-bit IP address of destination

- Like the addresses on an envelope

10

IP Fragmentation



- Every Network has Own Maximum Transmission Unit (MTU)
  - Largest IP datagram it can carry within its own packet frame
    - E.g., Ethernet is 1500 bytes
  - Don't know MTUs of all intermediate networks in advance
- IP Solution
  - When hit network with small MTU, fragment packets
    - Might get further fragmentation as proceed farther

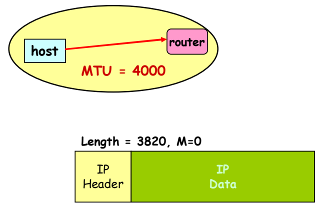
11

Fragmentation Related Fields

- Length
  - Length of IP fragment
- Identification
  - To match up with other fragments
- Fragment offset
  - Where this fragment lies in entire IP datagram
- Flags
  - "More fragments" flag
  - "Don't fragment" flag

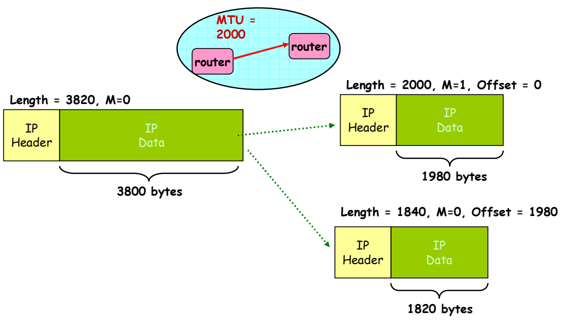
12

IP Fragmentation Example #1



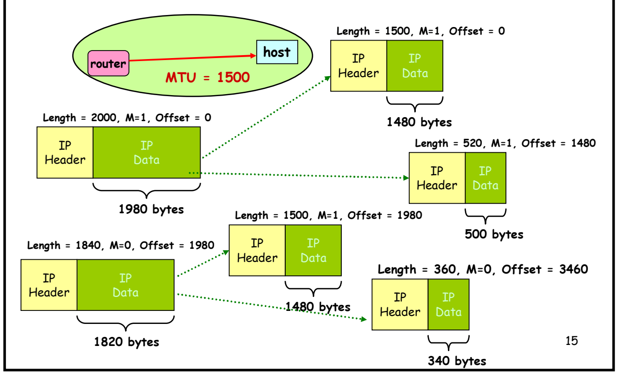
13

IP Fragmentation Example #2



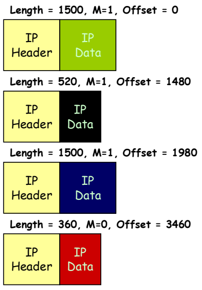
14

IP Fragmentation Example #3



15

IP Reassembly



- Fragments might arrive out-of-order
  - Don't know how much memory required until receive final fragment
- Some fragments may never arrive
  - After a while, give up entire process

16

Reassembly

- Where to do reassembly?
  - End nodes or at routers?
- End nodes -- better
  - Avoids unnecessary work where large packets are fragmented multiple times
  - If any fragment missing, delete entire packet
- Intermediate nodes -- Dangerous
  - How much buffer space required at routers?
  - What if routes in network change?
    - Multiple paths through network
    - All fragments only required to go through to destination

17

Fragmentation and Reassembly

- Demonstrates many Internet concepts
  - Decentralized
    - Every network can choose MTU
  - Connectionless
    - Each fragment contains full routing information
    - Fragments can proceed independently and along different routes
  - Complex endpoints and simple routers
    - Reassembly at endpoints
- Uses resources poorly
  - Forwarding, replication, encapsulations costs
  - Worst case: packet just bigger than MTU
  - Poor end-to-end performance
    - Loss of a fragment
- How to avoid fragmentation?
  - **Path MTU discovery protocol** → determines minimum MTU along route
  - Uses ICMP error messages

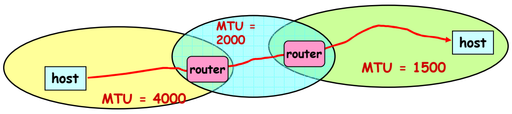
18

Internet Control Message Protocol (ICMP)

- Short messages used to send error & other control information
- Examples
  - Echo request / response
    - Can use to check whether remote host reachable
  - Destination unreachable
    - Indicates how far packet got & why couldn't go further
  - Flow control (source quench)
    - Slow down packet delivery rate
  - Timeout
    - Packet exceeded maximum hop limit
  - Router solicitation / advertisement
    - Helps newly connected host discover local router
  - Redirect
    - Suggest alternate routing path for future messages

19

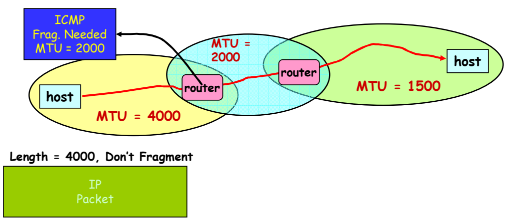
IP MTU Discovery with ICMP



- Operation
  - Send max-sized packet with "do not fragment" flag set
  - If encounters problem, ICMP message will be returned
    - "Destination unreachable: Fragmentation needed"
    - Usually indicates MTU encountered

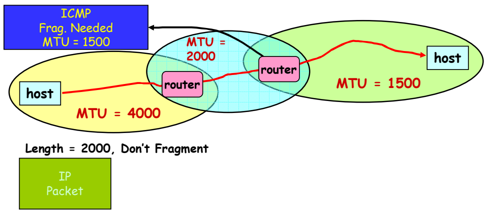
20

IP MTU Discovery with ICMP



21

IP MTU Discovery with ICMP

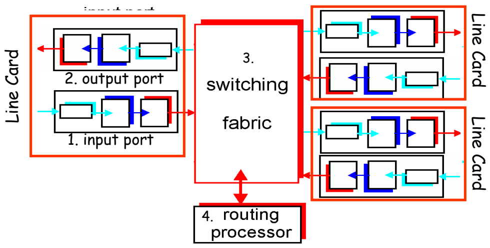


22

Router Architecture Overview

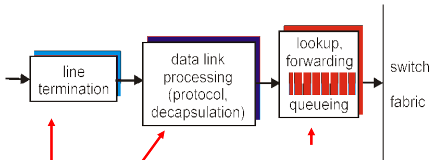
Two key router functions:

- Run routing algorithms/protocol (RIP, OSPF, BGP)
- *Switching* datagrams from incoming to outgoing link



23

Line Card: Input Port



Physical layer:
bit-level reception

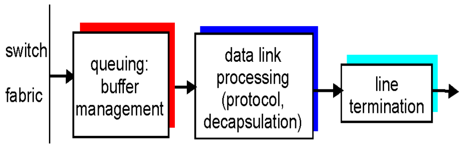
Data link layer:
e.g., Ethernet

Decentralized switching:

- Process common case ("fast-path") packets
  - Decrement TTL, update checksum, forward packet
- Given datagram dest., lookup output port using routing table in input port memory
- Queue needed if datagrams arrive faster than forwarding rate into switch fabric

24

Line Card: Output Port



- Queuing required when datagrams arrive from fabric faster than the line transmission rate

25

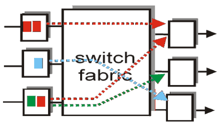
Buffering

- 3 types of buffering
  - Input buffering
    - Fabric slower than input ports combined → queuing may occur at input queues
      - Can avoid any input queuing by making switch speed = N x link speed
      - But need output buffering
  - Output buffering
    - Buffering when arrival rate via switch exceeds output line speed
  - Internal buffering
    - Can have buffering inside switch fabric to deal with limitations of fabric
- What happens when these buffers fill up?
  - Packets are THROWN AWAY!! This is where (most) packet loss comes from

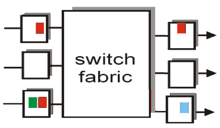
26

Input Port Queuing

- Which inputs are processed each slot – schedule?
- Head-of-the-Line (HOL) blocking: datagram at front of queue prevents others in queue from moving forward



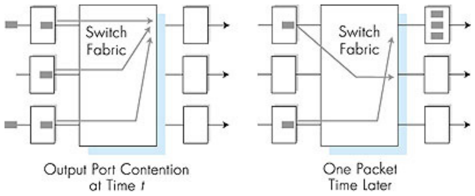
output port contention at time t - only one red packet can be transferred



green packet experiences HOL blocking

27

Output Port Queuing



- Scheduling discipline chooses among queued datagrams for transmission
  - Can be simple (e.g., first-come first-serve) or more clever (e.g., weighted round robin)

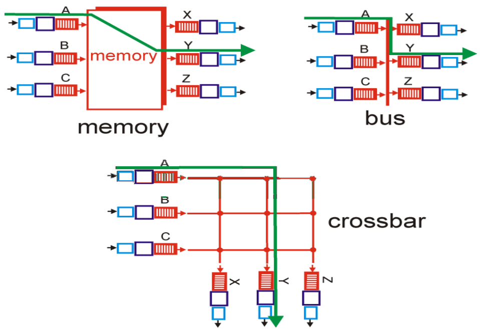
28

Network Processor

- Runs routing protocol and downloads forwarding table to forwarding engines
- Performs "slow" path processing
  - ICMP error messages
  - IP option processing
  - Fragmentation
  - Packets destined to router

29

Three Types of Switching Fabrics

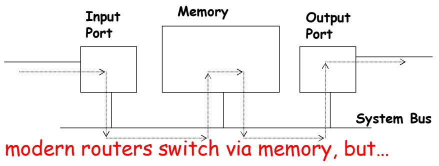


30

Switching Via a Memory

First generation routers → looked like PCs

- Packet copied by system's (single) CPU
- Speed limited by memory bandwidth (2 bus crossings per datagram)



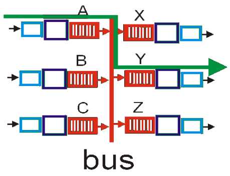
Most modern routers switch via memory, but…

- Input port processor performs lookup, copy into memory
- Cisco Catalyst 8500

31

Switching Via a Bus

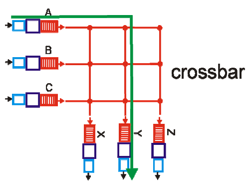
- Datagram from input port memory to output port memory via a shared bus
- Bus contention: switching speed limited by bus bandwidth
- 1 Gbps bus, Cisco 1900: sufficient speed for access and enterprise routers (not regional or backbone)



32

Switching Via an Interconnection Network

- Overcome bus and memory bandwidth limitations
- Crossbar provides full NxN interconnect
  - Expensive
  - Uses 2N buses
- Cisco 12000: switches Gbps through the interconnection network



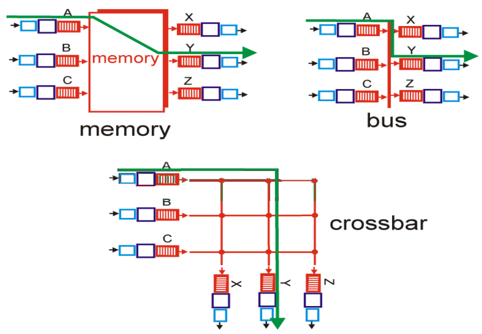
33

CS 640: Introduction to Computer Networks

Aditya Akella

Lecture 10 -
Intra-Domain Routing

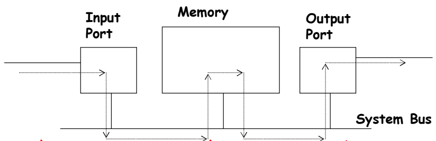
From previous lecture….
Three Types of Switching Fabrics



2

Switching Via a Memory

First generation routers → looked like PCs
- Packet copied by system's (single) CPU
- Speed limited by memory bandwidth (2 bus crossings per datagram)

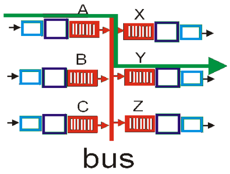


Most modern routers switch via memory, but…
- Input port processor performs lookup, copy into memory
- Cisco Catalyst 8500

3

Switching Via a Bus

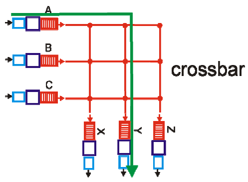
- Datagram from input port memory to output port memory via a shared bus
- Bus contention: switching speed limited by bus bandwidth
- 1 Gbps bus, Cisco 1900: sufficient speed for access and enterprise routers (not regional or backbone)



4

Switching Via an Interconnection Network

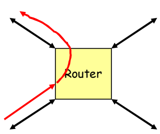
- Overcome bus and memory bandwidth limitations
- Crossbar provides full NxN interconnect
  - Expensive
  - Uses 2N buses
- Cisco 12000: switches Gbps through the interconnection network



5

Routing

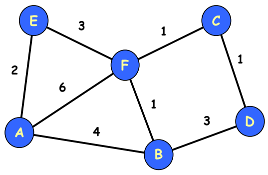
- Past two lectures
  - IP addresses are structured
  - IP packet headers carry these addresses
  - When packet arrives at router
    - Examine header for intended destination
    - Look up next hop in table
    - Send packet out appropriate port
- This lecture:
  - How these forwarding tables are built?
  - Routing algorithms



6

A Model of the Problem

- Network as a Graph:
  - Represent each router as node
  - Direct link between routers represented by edge
  - Symmetric links ⇒ undirected graph
- Edge "cost" c(x,y) denotes measure of difficulty of using link
  - delay, $ cost, or congestion level
- Task
  - Determine *least cost path* from every node to every other node
    - Path cost d(x,y) = sum of link costs



7

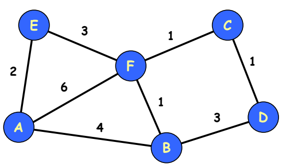
Ways to Compute Shortest Paths

- Centralized
  - Collect graph structure in one place
  - Use standard graph algorithm
  - Disseminate routing tables
- Distributed
  - Routers perform local computation
  - Converge to a globally consistent routing state
  - *"Global": Link-state*
    - Every node collects complete graph structure
    - Each computes shortest paths from it
    - Each generates own routing table
  - *Local: Distance-vector*
    - No one has copy of graph
    - Nodes construct their own tables iteratively
    - Each sends information about its table to neighbors

8

Distance-Vector Method

Initial Table for A		
Dest	Cost	Next Hop
A	0	A
B	4	B
C	∞	–
D	∞	–
E	2	E
F	6	F



- Idea
  - At any time, have cost/next hop of best known path to destination
  - Use cost ∞ when no path known
- Initially
  - Only have entries for directly connected nodes

9

Algorithm

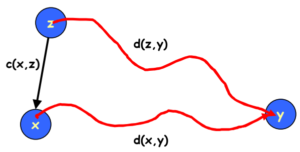
Each node x stores:
- c(x,v) for each neighbor v
- Distance vector of node x: estimate of d(x,y) for all y
- Distance vectors heard from each neighbor

Initialization:
1. d(x,y) = c(x,y) for all y.
2. Send distance vector to each neighbor

Repeat:
Whenever link cost to neighbor changes or distance vector received from neighbor
For every neighbor z
For every destination y
d(x,y) ← Update(x,y,z)

If d(x,y) changed for any y, send distance vector to all neighbors

Distance-Vector Update



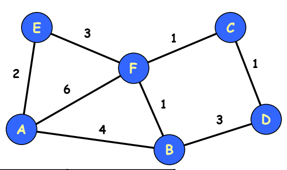
Update(x,y,z)
d ← c(x,z) + d(z,y) /* Cost of path from x to y with first hop z */
if d < d(x,y)
/* Found better path */
return d,z /* Updated cost / next hop */
else
return d(x,y), nexthop(x,y) /* Existing cost / next hop */

Start

Optimum 1-hop paths

Table for A			Table for B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	4	B	B	0	B
C	∞	-	C	∞	-
D	∞	-	D	3	D
E	2	E	E	∞	-
F	6	F	F	1	F

Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	∞	-	A	∞	-	A	2	A	A	6	A
B	∞	-	B	3	B	B	∞	-	B	1	B
C	0	C	C	1	C	C	∞	-	C	1	C
D	1	D	D	0	D	D	∞	-	D	∞	-
E	∞	-	E	∞	-	E	0	E	E	3	E
F	1	F	F	∞	-	F	3	F	F	0	F



Iteration #1

Optimum 2-hop paths

Table for A			Table for B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	4	B	B	0	B
C	7	F	C	2	F
D	7	B	D	3	D
E	2	E	E	4	F
F	5	E	F	1	F

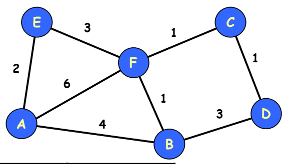


Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	7	F	A	7	B	A	2	A	A	5	B
B	2	F	B	3	B	B	4	F	B	1	B
C	0	C	C	1	C	C	4	F	C	1	C
D	1	D	D	0	D	D	∞	-	D	2	C
E	4	F	E	∞	-	E	0	E	E	3	E
F	1	F	F	2	C	F	3	F	F	0	F

13

Iteration #2

Optimum 3-hop paths

Table for A			Table for B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	4	B	B	0	B
C	6	E	C	2	F
D	7	B	D	3	D
E	2	E	E	4	F
F	5	E	F	1	F

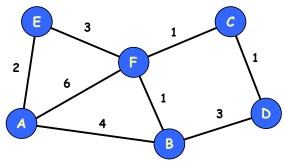


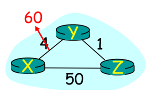
Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	6	F	A	7	B	A	2	A	A	5	B
B	2	F	B	3	B	B	4	F	B	1	B
C	0	C	C	1	C	C	4	F	C	1	C
D	1	D	D	0	D	D	5	F	D	2	C
E	4	F	E	5	C	E	0	E	E	3	E
F	1	F	F	2	C	F	3	F	F	0	F

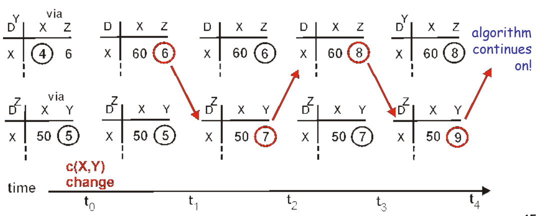
14

Distance Vector: Link Cost Changes

Link cost changes:

- Bad news travels slow - "count to infinity" problem!



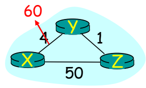


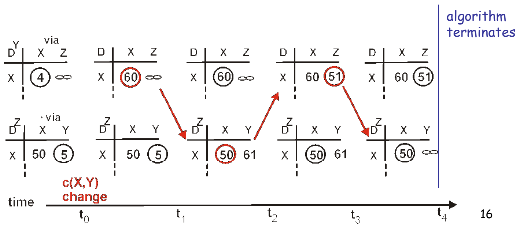
15

Distance Vector: Poison Reverse

If Z routes through Y to get to X :

- Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- Will this completely solve count to infinity problem?





16

Poison Reverse Failures

Table for A			Table for B			Table for D			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
C	7	F	C	8	A	C	9	B	C	1	C

Forced Update:

Table for A		
Dst	Cst	Hop
C	∞	-

Forced Update:

Table for F		
Dst	Cst	Hop
C	∞	-

Better Route:

Table for A		
Dst	Cst	Hop
C	13	D

Forced Update:

Table for B		
Dst	Cst	Hop
C	14	A

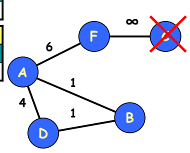
Forced Update:

Table for D		
Dst	Cst	Hop
C	15	B

Forced Update:

Table for A		
Dst	Cst	Hop
C	19	D

⋮



- Iterations don't converge
- "Count to infinity"
- Solution
  - Make "infinity" smaller
  - What is upper bound on maximum path length?

17

Routing Information Protocol (RIP)

- Earliest IP routing protocol (1982 BSD)
  - Current standard is version 2 (RFC 1723)
- Features
  - Every link has cost 1 → Hop count
  - "Infinity" = 16
    - Limits to networks where everything reachable within 15 hops
- Sending Updates
  - Every router listens for updates on UDP port 520
  - Triggered
    - When every entry changes, send copy of entry to neighbors
      - Except for one causing update (split horizon rule)
  - Periodic
    - Every 30 seconds, router sends copy of its table to each neighbor

18

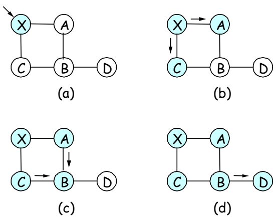
Link State Protocol Concept

- Every node gets complete copy of graph
  - Every node "floods" network with data about its outgoing links
- Every node computes routes to every other node
  - Using single-source, shortest-path algorithm
- Process performed whenever needed
  - When interconnections die / reappear

19

Sending Link States by "Flooding"

- X wants to send information
  - Sends on all outgoing links
- When node Y receives information from Z
  - Resend on all links other than Z



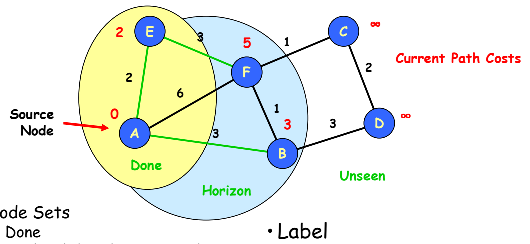
20

Dijkstra's Algorithm

- Given
  - Graph with source node s and edge costs c(u,v)
  - Determine least cost path from s to every node v
- Single source shortest Path Algorithm
  - Traverse graph in order of least cost from source

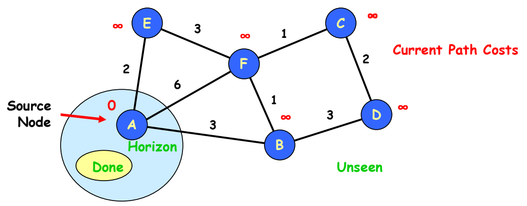
21

Dijkstra's Algorithm



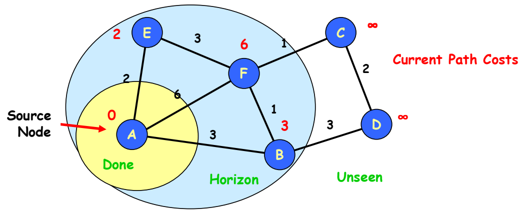
- Node Sets
  - Done
    - Already have least cost path to it
  - Horizon:
    - Reachable in 1 hop from node in Done
  - Unseen:
    - Cannot reach directly from node in Done
- Label
  - d(v) = path cost
    - From s to v
- Path
  - Keep track of last link in path

Dijkstra's Algorithm: Initially



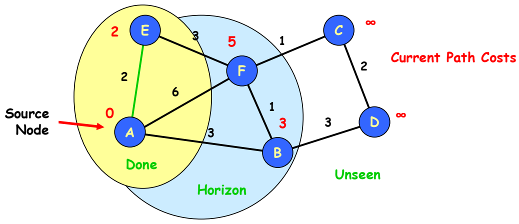
- No nodes "done"
- Source in "horizon"

Dijkstra's Algorithm: Initially



- d(v) to node A shown in red
  - Only consider links from done nodes

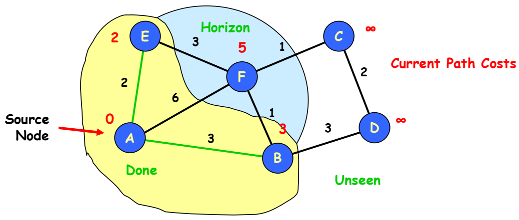
Dijkstra's Algorithm



- Select node v in horizon with minimum d(v)
- Add link used to add node to shortest path tree
- Update d(v) information

25

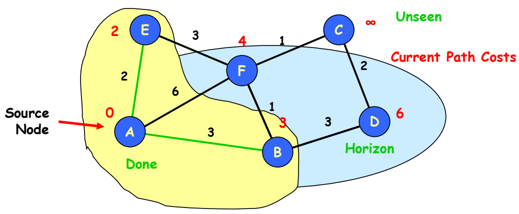
Dijkstra's Algorithm



- Repeat…

26

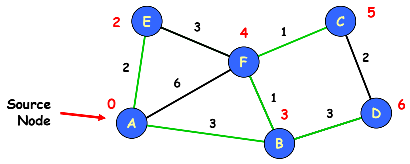
Dijkstra's Algorithm



- Addition of node can add new nodes to horizon

27

Dijkstra's Algorithm

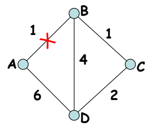


- Final tree shown in green

28

Link State Characteristics

- With consistent LSDBs*, all nodes compute consistent loop-free paths
- Can still have transient loops



Packet from C→A may loop around BDC if B knows about failure and C & D do not

*Link State Data Base

29

OSPF Routing Protocol

- Open
  - Open standard created by IETF
- More prevalent than RIP

30

OSPF Messages

- Transmit link state advertisements
  - Originating router
    - Typically, IP address for router
  - Link ID
    - ID of router at other end of link
  - Metric
    - Cost of link
  - Sequence number
    - Incremented each time sending new link information

31

OSPF Flooding Operation

- Node X Receives LSA from Node Y
  - With Sequence Number q
  - Looks for entry with same origin/link ID
- Cases
  - No entry present
    - Add entry, propagate to all neighbors other than Y
  - Entry present with sequence number $p < q$
    - Update entry, propagate to all neighbors other than Y
  - Entry present with sequence number $p > q$
    - Send entry back to Y
    - To tell Y that it has out-of-date information
  - Entry present with sequence number $p = q$
    - Ignore it

32

Flooding Issues

- When should it be performed
  - Periodically
  - When status of link changes
    - Detected by connected node
    - Congestion, lack of electric or optical signal
- What happens when router goes down & back up
  - Sequence number reset to 0
    - Other routers may have entries with higher sequence numbers
  - Router will send out LSAs with number 0
  - Will get back LSAs with last valid sequence number p
  - Router sets sequence number to p+1 & resends

33

Adoption of OSPF

- RIP viewed as outmoded
  - Good when networks small and routers had limited memory & computational power
- OSPF Advantages
  - Fast convergence when configuration changes
  - Full topology map helps

34

Comparison of LS and DV Algorithms

Message complexity
- LS: with n nodes, v neighbors, O(nv) messages per node
- DV: exchange between neighbors only

Speed of Convergence
- LS: Complex computation
  - But…can forward before computation
  - may have oscillations
- DV: convergence time varies
  - may be routing loops
  - count-to-infinity problem
  - (faster with triggered updates)

35

Comparison of LS and DV Algorithms

Robustness: what happens if router malfunctions?

LS:
- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:
- DV node can advertise incorrect *path* cost
- each node's table used by others
  - errors propagate thru network

- Other tradeoffs
  - Making LSP flood reliable difficult
  - Prioritize routing packets?

36

CS 640: Introduction to Computer Networks

Aditya Akella

Lecture 11 -
Inter-Domain Routing -
BGP (Border Gateway Protocol)

Intra-domain routing

- The Story So Far…
  - Routing protocols generate the forwarding table
  - Two styles: distance vector, link state
  - Scalability issues:
    - Distance vector protocols suffer from count-to-infinity
    - Link state protocols must flood information through network
- Today's lecture
  - How to make routing protocols support large networks
  - How to make routing protocols support business policies

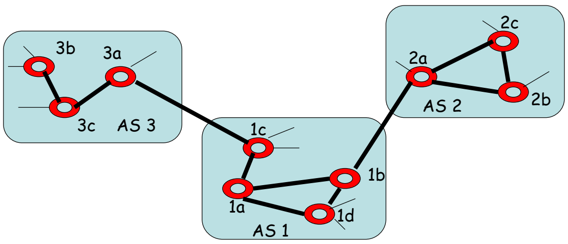
Inter-domain Routing: Hierarchy

- "Flat" routing not suited for the Internet
  - Doesn't scale with network size
    - Storage → Each node cannot be expected to store routes to every destination (or destination network)
    - Convergence times increase
    - Communication → Total message count increases
  - Administrative autonomy
    - Each internetwork may want to run its network independently
      - E.g hide topology information from competitors
- Solution: Hierarchy via autonomous systems

Internet's Hierarchy

- What is an Autonomous System (AS)?
  - A set of routers under a single technical administration
    - Use an *interior gateway protocol (IGP)* and common metrics to route packets within the AS
    - Connect to other ASes using *gateway routers*
    - Use an *exterior gateway protocol (EGP)* to route packets to other AS's
  - IGP: OSPF, RIP (last class)
  - Today's EGP: BGP version 4
  - Similar to an "inter-network"
    - Could also be a group of internetworks owned by a single commercial entity

An example



Intra-AS routing algorithm + Inter-AS routing algorithm → Forwarding table

The Problem

- Easy when only one link leading to outside AS
- Much harder when two or more links to outside ASes
  - Which destinations reachable via a neighbor?
  - Propagate this information to other internal routers
  - Select a "good route" from multiple choices
  - Inter-AS routing protocol
    - Communication between distinct ASes
    - Must be the same protocol!

BGP Preliminaries

- Pairs of routers exchange routing info over TCP connections (port 179)
  - One TCP connection for every pair of neighboring gateway routers
  - Routers called "BGP peers"
  - BGP peers exchange routing info as messages
  - TCP connection + messages → BGP session
- Neighbor ASes exchange info on which CIDR prefixes are reachable via them
- Primary objective: reachability not performance

7

AS Numbers (ASNs)

ASNs are 16 bit values   64512 through 65535 are "private"

Currently over 15,000 in use

- Genuity: 1
- MIT: 3
- CMU: 9
- UC San Diego: 7377
- AT&T: 7018, 6341, 5074, …
- UUNET: 701, 702, 284, 12199, …
- Sprint: 1239, 1240, 6211, 6242, …
- …

ASNs represent units of routing policy

8

Distance Vector with Path

- Each routing update carries the entire AS-level path so far
  - "AS_Path attribute"
- Loops are detected as follows:
  - When AS gets route, check if AS already in path
    - If yes, reject route
    - If no, add self and (possibly) advertise route further
      - Advertisement depends on metrics/cost/preference etc.
- Advantage:
  - Metrics are local - AS chooses path, protocol ensures no loops

9

Hop-by-hop Model

- BGP advertises to neighbors only those routes that it uses
  - Consistent with the hop-by-hop Internet paradigm
  - Consequence: hear only one route from neighbor
    - (although neighbor may have chosen this from a large set of choices)
    - Could impact view into availability of paths

10

Policy with BGP

- BGP provides capability for enforcing various policies
- Policies are **not** part of BGP: they are provided to BGP as configuration information
- **Enforces** policies by
  - *Choosing appropriate paths* from multiple alternatives
  - *Controlling advertisement* to other AS's

11

Examples of BGP Policies

- A multi-homed AS refuses to act as transit
  - Limit path advertisement
- A multi-homed AS can become transit for some AS's
  - Only advertise paths to some AS's
- An AS can favor or disfavor certain AS's for traffic transit from itself

12

BGP Messages

- Open
  - Announces AS ID
  - Determines hold timer – interval between keep_alive or update messages, zero interval implies no keep_alive
- Keep_alive
  - Sent periodically (but before hold timer expires) to peers to ensure connectivity.
  - Sent in place of an UPDATE message
- Notification
  - Used for error notification
  - TCP connection is closed *immediately* after notification

13

BGP UPDATE Message

- List of withdrawn routes
- Network layer reachability information
  - List of reachable prefixes
- Path attributes
  - Origin
  - Path
  - Local_pref → this is set locally
  - MED → this is set externally
  - Metrics
- All prefixes advertised in message have same path attributes

14

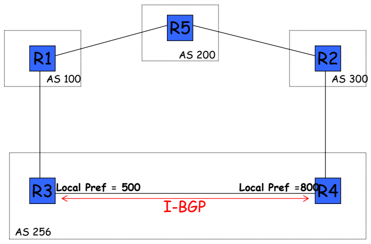
Path Selection Criteria

- Attributes + external (policy) information
- Examples:
  - Policy considerations
    - Preference for AS
    - Presence or absence of certain AS
  - Hop count
  - Path origin

15

LOCAL PREF

- Local (within an AS) mechanism to provide relative priority among BGP exit points

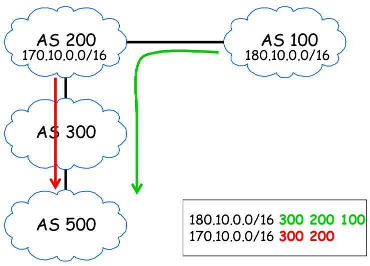


- Prefer routers announced by one AS over another or general preference over routes

16

AS_PATH

- List of traversed AS's



17

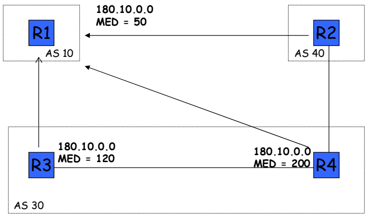
Multi-Exit Discriminator (MED)

- Hint to external neighbors about the preferred path *into* an AS
  - Different AS choose different scales
- Used when two AS's connect to each other in more than one place
  - More useful in a customer provider setting
  - Not honored in other settings
    - Will see later why

18

MED

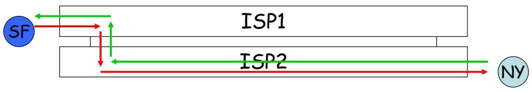
- Hint to R1 to use R3 over R4 link
- Cannot compare AS40's values to AS30's



19

MED

- MED is typically used in provider/subscriber scenarios
- It can lead to unfairness if used between ISP because it may force one ISP to carry more traffic:



- ISP1 ignores MED from ISP2
- ISP2 obeys MED from ISP1
- ISP2 ends up carrying traffic most of the way

20

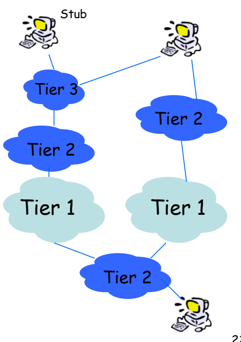
Decision Process (First cut)

- Rough processing order of attributes:
  - Select route with highest LOCAL-PREF
  - Select route with shortest AS-PATH
  - Apply MED (to routes learned from same neighbor)
- How to set the attributes?
  - Especially local_pref?
  - Policies in action

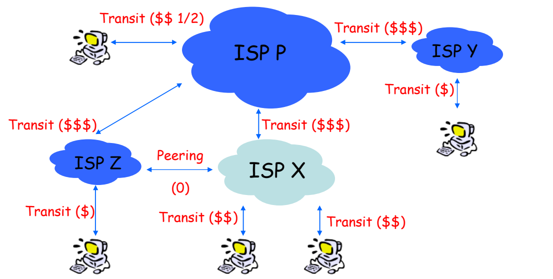
21

A Logical View of the Internet

- Tier 1 ISP
  - "Default-free" with global reachability info
- Tier 2 ISP
  - Regional or country-wide
  - Typically route through tier-1
    - Customer
- Tier 3/4 ISPs
  - Local
  - Route through higher tiers
- Stub AS
  - End network such as IBM or UW-Madison

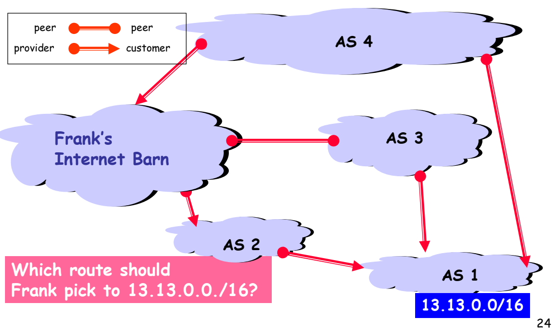


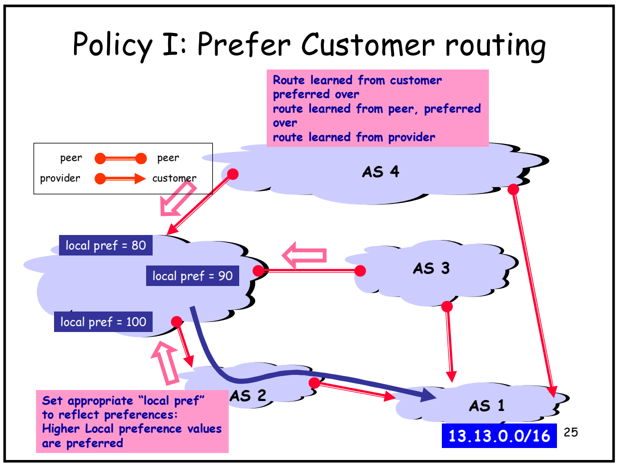
Inter-ISP Relationships: Transit vs. Peering

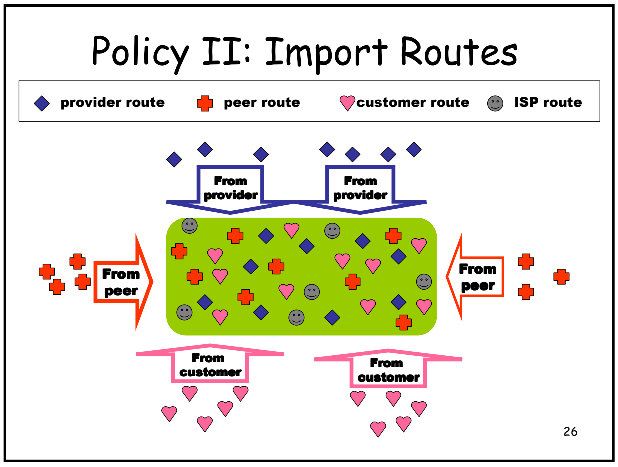


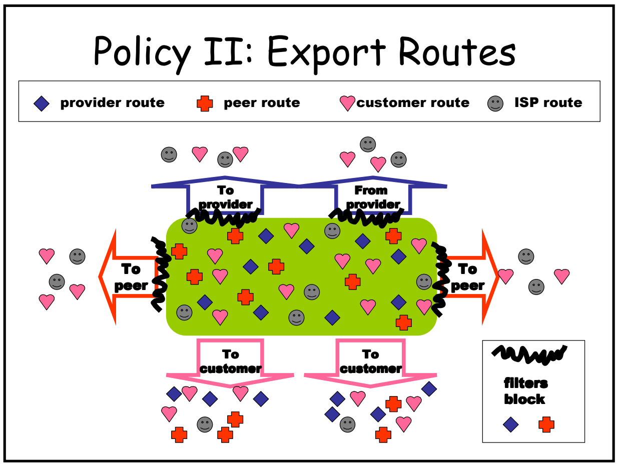
These relationships have the greatest impact on BGP policies

Illustrating BGP Policies




Policy I: Prefer Customer routing
Route learned from customer preferred over route learned from peer, preferred over route learned from provider
peer peer
provider customer
AS 4
local pref = 80
local pref = 90
AS 3
local pref = 100
AS 2
AS 1
Set appropriate "local pref" to reflect preferences: Higher Local preference values are preferred
13.13.0.0/16
25


Policy II: Import Routes
provider route
peer route
customer route
ISP route
From provider
From provider
From peer
From peer
From customer
From customer
26

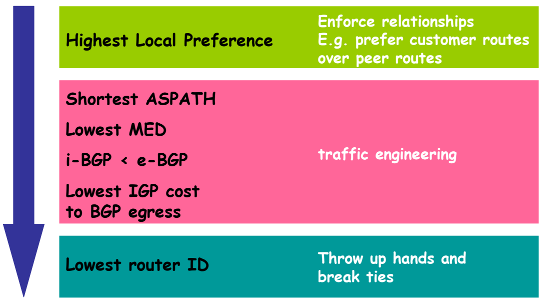

Policy II: Export Routes
provider route
peer route
customer route
ISP route
To provider
From provider
To peer
To peer
To customer
To customer
filters block

Policy II: Valley-Free Routes

- "Valley-free" routing
  - Number links as (+1, 0, -1) for provider, peer and customer
  - In any *valid* path should only see sequence of +1, followed by at most one 0, followed by sequence of -1
  - Why?
    - Consider the economics of the situation
- How to make these choices?
  - Prefer-customer routing: LOCAL_PREF
  - Valley-free routes: control route advertisements (see previous slide)

28

BGP Route Selection Summary



29

CS 640: Introduction to Computer Networks

Aditya Akella

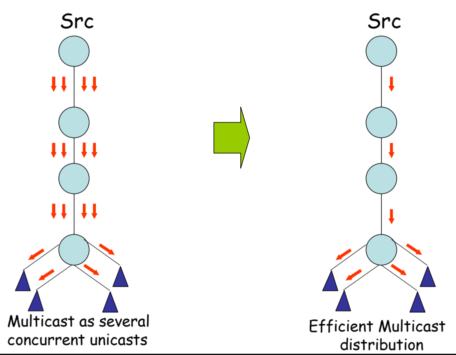
Lecture 12 - Multicast

Multicast

- Unicast: one source to one destination
  - Web, telnet, FTP, ssh
- Broadcast: one source to all destinations
  - Never used over the Internet
  - LAN applications
- *Multicast*: one source to many destinations
  - Several important applications
- Multicast goal: efficient data distribution

2

Multicast – Efficient Data Distribution



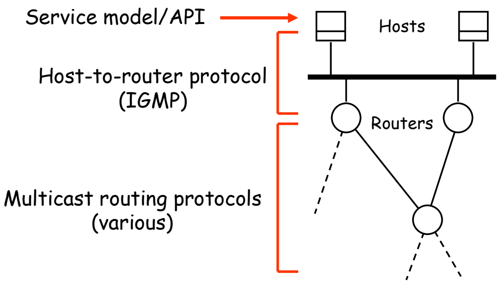
3

Multicast Example Applications

- Broadcast audio/video
- Push-based systems
- Software distribution
- Teleconferencing (audio, video, shared whiteboard, text editor)
- Multi-player games
- Server/service location
- Other distributed applications

4

IP Multicast Architecture



5

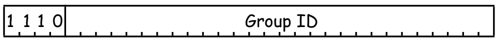
IP Multicast Service Model (rfc1112)

- Each group identified by a single IP address
- Groups may be of any size
- Members of groups may be located anywhere in the Internet
  - We will focus on an internetwork
- Members of groups can join and leave at will
- Senders need not be members
- Group membership not known explicitly

6

IP Multicast Addresses

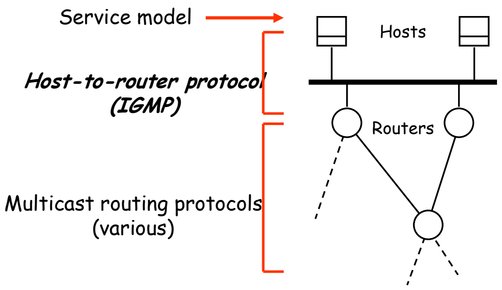
- Class D IP addresses
  - 224.0.0.0 - 239.255.255.255



- How to allocate these addresses?
  - Well-known multicast addresses, assigned by IANA
  - Transient multicast addresses, assigned and reclaimed dynamically
    - e.g., by "sdr" program
- Interested recipients must *join* a group by selecting the appropriate multicast group address

7

IP Multicast Architecture



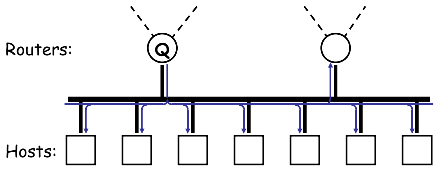
8

Internet Group Management Protocol

- End system to router protocol is IGMP
- Each host keeps track of which mcast groups it has subscribed to
  - Socket API informs IGMP process of all joins
- Objective is to keep router up-to-date with group membership of entire LAN
  - Routers need not know who all the members are, only that *members exist*

9

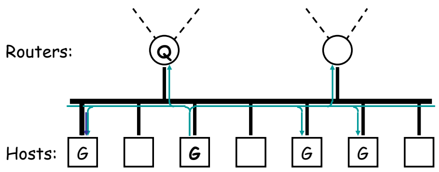
How IGMP Works



- On each link, one router is elected the "querier"
- Querier periodically sends a Membership Query message to the all-systems group (224.0.0.1), with TTL = 1
- On receipt, hosts start random timers (between 0 and 10 seconds) for each multicast group to which they belong

10

How IGMP Works (cont.)



- When a host's timer for group G expires, it sends a Membership Report <u>to group G</u>, with TTL = 1
- Other members of G hear the report and stop their timers
- Routers hear <u>all</u> reports, and time out non-responding groups
  - "Soft state" again

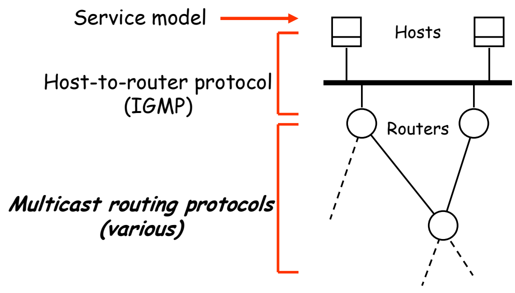
11

How IGMP Works (cont.)

- Note that, in normal case, only one report message per group present is sent in response to a query
- Query interval is typically 60-90 seconds
- When a host first joins a group, it sends one or two immediate reports, instead of waiting for a query

12

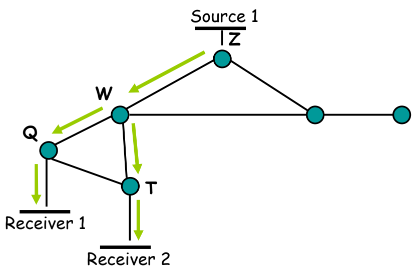
IP Multicast Architecture



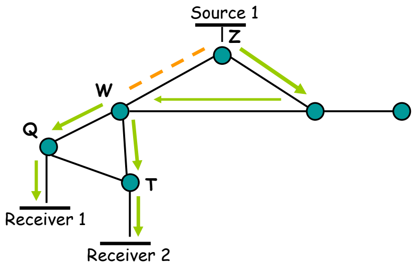
Routing Techniques

- Basic objective – routers must collectively build distribution tree for multicast packets
- Flood and prune based approach for DV-networks
  - Begin by flooding traffic to entire network
  - Prune branches with no receivers
  - Examples: DVMRP
- Link-state based networks use a different approach
  - Routers advertise groups for which they have receivers to entire network
  - Compute trees on demand
  - Example: MOSPF
- There are several others: PIM-SM, PIM-DM, CBT…
  - These are "rendezvous-based" approaches
  - Independent of underlying routing protocol

MOSPF: Example



Link Failure/Topology Change



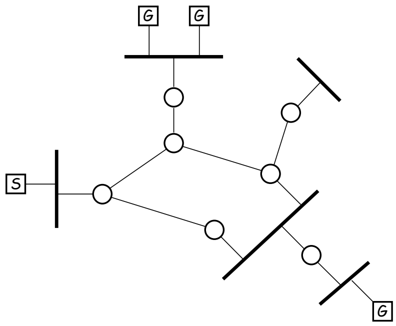
Impact on Route Computation

- Hard to pre-compute multicast trees for all possible sources and all possible groups
  - Otherwise, may end up with a lot of unwanted state where there are no senders
- Compute on demand when first packet from a source S to a group G arrives
- New link-state advertisement
  - May lead to addition or deletion of outgoing interfaces if it contains different group addresses
  - May lead to re-computation of entire tree if links are changed

Distance-Vector Multicast Routing

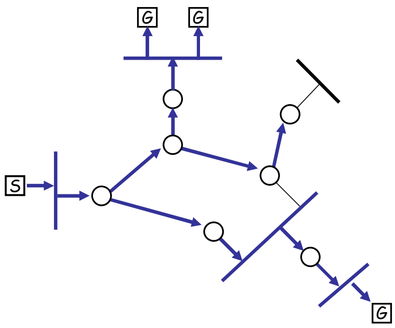
- DVMRP consists of two major components:
  - A conventional distance-vector routing protocol (like RIP)
  - A protocol for determining how to forward multicast packets, based on the routing table
- DVMRP router forwards a packet if
  - The packet arrived from the link used to reach the source of the packet (reverse path forwarding check – RPF)
  - If downstream links have *not pruned* the tree

Example Topology



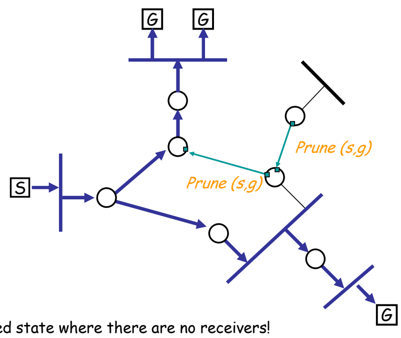
19

Broadcast with Truncation



20

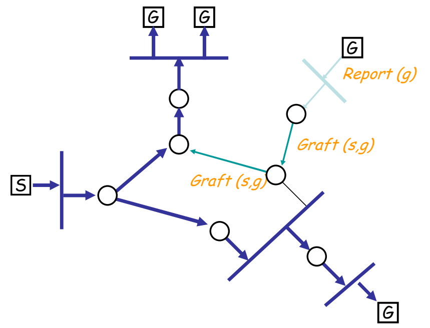
Prune



Unwanted state where there are no receivers!

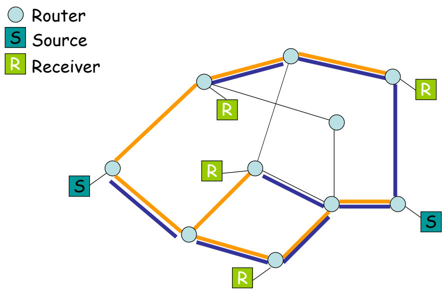
21

Graft



22

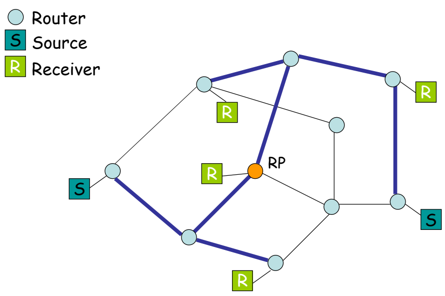
Source-based Trees



Both protocols discussed today use this approach

23

Shared Tree



24

Shared vs. Source-Based Trees

- Source-based trees
  - Shortest path trees – low delay, better load distribution
  - More state at routers (per-source state)
  - Efficient for *dense-area multicast*
- Shared trees
  - Higher delay (bounded by factor of 2), traffic concentration
  - Choice of core affects efficiency
  - Per-group state at routers
  - Efficient for *sparse-area multicast:* PIM-SM

25