Computer Networks (CS422)

Douglas Comer

Computer Science Department
Purdue University
West Lafayette, IN 47907

http://www.cs.purdue.edu/people/comer

NOTES

COMPUTER NETWORKS
(CS 422)

2003

Professor Douglas Comer

PART I

Introduction

Topic And Scope

Computer networks and internets: an overview of concepts, terminology, and technologies that form the basis for digital communication in private corporate networks and the global Internet

NOTES

You Will Learn

- Terminology
- Communication basics
  - Media and signals
  - Asynchronous and synchronous communication
  - Relationships among bandwidth, throughput, and noise
  - Frequency-division and time-division multiplexing

You Will Learn
(continued)

- Networking and network technologies
  - Packet switching
  - Framing, parity, and error detection
  - Local and wide area technologies
  - Network addressing
  - Connection and extension (repeaters, bridges, hubs, switches)
  - Topologies and wiring (star, ring, bus)
  - Next-hop forwarding
  - Shortest path computation
  - Measures of delay and throughput
  - Protocol layers

NOTES

You Will Learn
(continued)

- Internets and Internetworking
  - Motivation and concept
  - Internet Protocol (IP) datagram format and addressing
  - Internet routers and routing
  - Address binding (ARP)
  - Internet control messages (ICMP)
  - User Datagram Protocol (UDP)
  - Transmission Control Protocol (TCP)
  - Protocol ports and demultiplexing

You Will Learn
(continued)

- Network applications
  - Client-server paradigm
  - Domain name system (DNS)
  - File transfer (FTP)
  - Mail transfer (SMTP)
  - IP Telephony
  - Remote login (TELNET)
  - Web technologies and protocols (HTTP, CGI)
  - Network security

NOTES

What You Will Not Learn

- Commercial aspects
  - Products
  - Vendors
  - Prices
  - Network operating systems
- How to purchase/configure/operate
- How to design/implement protocol software

Background Required

- Ability to program in C
- Knowledge of low-level programming constructs
  - Pointers
  - Bit fields in structures
  - Printf
- Familiarity with basic tools
  - Text editor
  - Compiler/linker/loader

NOTES

Background Required
(continued)

- Basic knowledge of operating systems
  - Terminology
  - Functionality
  - Processes and concurrent processing
- Desire to learn

Schedule Of Topics

- Signals, media, bandwidth, throughput, and multiplexing (~1 weeks)
- Networking: concepts, technologies (~4 weeks)
- Internetworking fundamentals (~5 weeks)
- Internet applications (~5 weeks)

NOTES

Labs

- An essential part of the course
- You will gain hands-on experience with
  - Network programming and applications
  - Packet analysis
  - Network measurement
  - Socket programming
- Labs will start with network applications right away!

Motivation For Networking

- Information access
- Interaction among cooperative application programs
- Resource sharing

NOTES

Practical Results

- E-mail
- File transfer/access
- Web browsing
- Remote login/execution
- IP Telephony
- The Internet

What A Network Includes

- Transmission hardware
- Special-purpose hardware devices
  - Interconnect transmission media
  - Control transmission
  - Run protocol software
- Protocol software
  - Encodes and formats data
  - Detects and corrects problems

NOTES

What A Network Does

- Provides communication that is
  - Reliable
  - Fair
  - Efficient
  - Secure
  - From one application to another

What A Network Does
(continued)

- Automatically detects and corrects
  - Data corruption
  - Data loss
  - Duplication
  - Out-of-order delivery
- Automatically finds optimal path from source to destination

NOTES

Network Programming

- Network allows arbitrary applications to communicate
- Programmer does not need to understand network technologies
- Network facilities accessed through an *Application Program Interface*

Basic Paradigm For Internet Communication

- Establish contact
- Exchange data (bi-directional)
- Terminate contact

NOTES

Establishing Contact

- Performed by pair of applications
- One application starts and waits for contact (called *server*)
- Other application initiates contact (called *client*)

Identifying A Waiting Application

- Conceptually two items specified
  - Computer
  - Application on that computer
- Terminology
  - Computer identified by *domain name*
  - Application identified by program name

NOTES

Representations And Translations

- Humans use names such as
  - *www.netbook.cs.purdue.edu* (computer)
  - *ftp* (application)
- Network protocols require binary values
- Library routines exist to translate from names to numbers

Example API

Operation	Meaning
await_contact	used by a server to wait for contact from a client
make_contact	used by a client to contact a server
cname_to_comp	used to translate a computer name to an equivalent internal binary value
appname_to_appnum	used to translate a program name to an equivalent internal binary value
send	used by either client or server to send data
recv	used by either client or server to receive data
send_eof	used by both client and server after they have finished sending data

NOTES

Example #1: Echo

- Useful for network testing
- Server returns exact copy of data sent
- User on computer *X* runs

```
echoserver 22000
```

- User on another computer runs

```
echoclient X 22000
```

Example #2: Chat

- Miniature version of Internet chat service
- Allows two users to communicate
- User on computer *X* runs

```
chatserver  25000
```

- User on another computer runs

```
chatclient X 25000
```


NOTES

Example Application: Web Server

- User on computer *X* runs

```
webserver  27000
```

- User on another computer runs browser and enters URL:

http://*X*:27000/index.html

Example Code Using API: Echoserver

```
/* echoserver.c */

#include <stdlib.h>
#include <stdio.h>
#include <cnaiapi.h>
#include "win32.h"

#define BUFFSIZE                256

/*------------------------------------------------------------------------
 *
 * Program: echoserver
 * Purpose: wait for a connection from an echoclient and echo data
 * Usage:   echoserver <appnum>
 *
 *------------------------------------------------------------------------
 */
int
main(int argc, char *argv[])
{
        connection      conn;
        int             len;
        char            buff[BUFFSIZE];

        if (argc != 2) {
                (void) fprintf(stderr, "usage: %s <appnum>\n", argv[0]);
                exit(1);
        }
```


NOTES

Echoserver (2 of 2)

```
        /* wait for a connection from an echo client */

        conn = await_contact((appnum) atoi(argv[1]));
        if (conn < 0)
                exit(1);

        /* iterate, echoing all data received until end of file */

        while((len = recv(conn, buff, BUFFSIZE, 0)) > 0)
                (void) send(conn, buff, len, 0);
        send_eof(conn);
        return 0;
}
```

- Actually works on the Internet
- API calls replace conventional I/O
- No networking knowledge required

Example Code Using API: Webserver

```
/* webserver.c */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <cnaiapi.h>
#include "win32.h"

#if defined(LINUX) || defined(SOLARIS)
#include <sys/time.h>
#endif

#define BUFFSIZE        256
#define SERVER_NAME     "CNAI Demo Web Server"

#define ERROR_400       "<head></head><body><html><h1>Error 400</h1><p>The server \
couldn't understand your request.</html></body>\n"

#define ERROR_404       "<head></head><body><html><h1>Error 404</h1><p>Document \
not found.</html></body>\n"

#define HOME_PAGE       "<head></head><body><html><h1>Welcome to the CNAI Demo \
Server</h1><p>Why not visit: <ul><li><a href=\"http://netbook.cs.purdue.edu\
\">Netbook Home Page</a><li><a href=\"http://www.comerbooks.com\">Comer \
Books Home Page<a></ul></html></body>\n"

#define TIME_PAGE       "<head></head><body><html><h1>The current date is: %s</h1>\
</html></body>\n"

int     recvln(connection, char *, int);
```


NOTES

Webserver (2 of 6)

```
/*------------------------------------------------------------------------
 *
 * Program: webserver
 * Purpose: serve hard-coded webpages to web clients
 * Usage:   webserver <appnum>
 *
 *------------------------------------------------------------------------
 */
int
main(int argc, char *argv[])
{

        connection      conn;
        int             n;
        char            buff[BUFFSIZE], cmd[16], path[64], vers[16];
        char            *timestr;
#if defined(LINUX) || defined(SOLARIS)
        struct timeval  tv;
#elif defined(WIN32)
        time_t          tv;
#endif

        if (argc != 2) {
                (void) fprintf(stderr, "usage: %s <appnum>\n", argv[0]);
                exit(1);
        }
```

Webserver (3 of 6)

```
        while(1) {

                /* wait for contact from a client on specified appnum */

                conn = await_contact((appnum) atoi(argv[1]));
                if (conn < 0)
                        exit(1);

                /* read and parse the request line */

                n = recvln(conn, buff, BUFFSIZE);
                sscanf(buff, "%s %s %s", cmd, path, vers);

                /* skip all headers - read until we get \r\n alone */

                while((n = recvln(conn, buff, BUFFSIZE)) > 0) {
                        if (n == 2 && buff[0] == '\r' && buff[1] == '\n')
                                break;
                }

                /* check for unexpected end of file */

                if (n < 1) {
                        (void) send_eof(conn);
                        continue;
                }
```


NOTES

Webserver (4 of 6)

```
                /* check for a request that we cannot understand */

                if (strcmp(cmd, "GET") || (strcmp(vers, "HTTP/1.0") &&
                                           strcmp(vers, "HTTP/1.1"))) {
                        send_head(conn, 400, strlen(ERROR_400));
                        (void) send(conn, ERROR_400, strlen(ERROR_400),0);
                        (void) send_eof(conn);
                        continue;
                }

                /* send the requested web page or a "not found" error */

                if (strcmp(path, "/") == 0) {
                        send_head(conn, 200, strlen(HOME_PAGE));
                        (void) send(conn, HOME_PAGE, strlen(HOME_PAGE),0);
                } else if (strcmp(path, "/time") == 0) {
#if defined(LINUX) || defined(SOLARIS)
                        gettimeofday(&tv, NULL);
                        timestr = ctime(&tv.tv_sec);
#endif
                        (void) sprintf(buff, TIME_PAGE, timestr);
                        send_head(conn, 200, strlen(buff));
                        (void) send(conn, buff, strlen(buff), 0);
                } else { /* not found */
                        send_head(conn, 404, strlen(ERROR_404));
                        (void) send(conn, ERROR_404, strlen(ERROR_404),0);
                }
                (void) send_eof(conn);
        }
}
```

Webserver (5 of 6)

```
/*------------------------------------------------------------------------
 * send_head - send an HTTP 1.0 header with given status and content-len
 *------------------------------------------------------------------------
 */
void
send_head(connection conn, int stat, int len)
{
        char    *statstr, buff[BUFFSIZE];

        /* convert the status code to a string */

        switch(stat) {
        case 200:
                statstr = "OK";
                break;
        case 400:
                statstr = "Bad Request";
                break;
        case 404:
                statstr = "Not Found";
                break;
        default:
                statstr = "Unknown";
                break;
        }
```


NOTES

Webserver (6 of 6)

```
        /*
         * send an HTTP/1.0 response with Server, Content-Length,
         * and Content-Type headers.
         */

        (void) sprintf(buff, "HTTP/1.0 %d %s\r\n", stat, statstr);
        (void) send(conn, buff, strlen(buff), 0);

        (void) sprintf(buff, "Server: %s\r\n", SERVER_NAME);
        (void) send(conn, buff, strlen(buff), 0);

        (void) sprintf(buff, "Content-Length: %d\r\n", len);
        (void) send(conn, buff, strlen(buff), 0);

        (void) sprintf(buff, "Content-Type: text/html\r\n");
        (void) send(conn, buff, strlen(buff), 0);

        (void) sprintf(buff, "\r\n");
        (void) send(conn, buff, strlen(buff), 0);
}
```

Summary

- Studying networks is important because
  - The world is interconnected
  - Applications now operate in a distributed environment
- This course
  - Covers *all* of networking and internetworking
  - Explains the mystery
  - Will be hard work

NOTES

Summary
(continued)

- Computer networks
  - Deliver data from source to destination
  - Automatically find optimal paths
  - Handle problems that occur
- We will learn how networks do the above

NOTES

PART II

Signals, Media, and Data Transmission

Transmission Of Information

- Well-understood basics
- From physics
  - Energy
  - Electromagnetic wave propagation
- From mathematics
  - Coding theory

Transmission Media

- Copper wire
  - Need two wires
  - Possibilities
    - Twisted pair
    - Coaxial cable
- Optical fiber
  - Flexible
  - Light ‘‘stays in’’
- Air/space
  - Used for electromagnetic transmission

NOTES

Forms Of Energy Used To Transmit Data

- Electric current
- Audible sounds
- Omni-directional electromagnetic waves
  - Radio Frequency (RF)
  - Infrared

Forms Of Energy Used To Transmit Data (continued)

- Directional electromagnetic waves
  - Point-to-point satellite channel
  - Limited broadcast (spot beam)
  - Microwave
  - Laser beam

NOTES

Types Of Satellites

- Geosynchronous Earth Orbit (GEO)
- Low Earth Orbit (LEO)
  - Array needed

Two Important Physical Limits Of A Transmission System

- *Propagation delay*
  - Time required for signal to travel across media
  - Example: electromagnetic radiation travels through space at the speed of light ($C = 3 \times 10^8$ meters per second)
- *Bandwidth*
  - Maximum times per second the signal can change

NOTES

Transmission Of Data

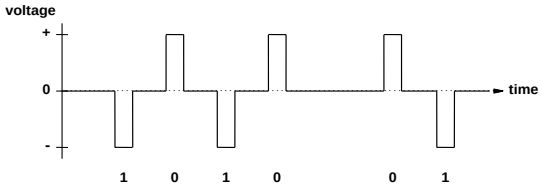
- Network hardware encodes information for transmission
- Two types of encoding
  - Analog (amount of energy proportional to value of item sent)
  - Digital (two forms of energy to encode *0* and *1*)
- Computer networks use the latter

Example Digital Encoding

- Medium
  - Copper wire
- Energy form
  - Electric current
- Encoding
  - Negative voltage encodes *1*
  - Positive voltage encodes *0*

NOTES

Illustration Of Digital Encoding



- Known as *waveform diagram*
- X-axis corresponds to time
- Y-axis corresponds to voltage

Encoding Details

- All details specified by a *standard*
- Several organizations produce networking standards
  - IEEE
  - ITU
  - EIA
- Hardware that adheres to standard interoperable

NOTES

The RS-232C Standard

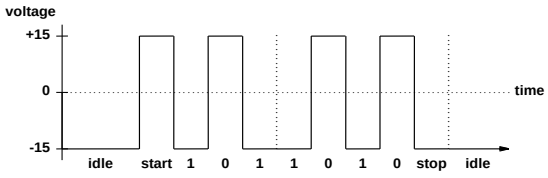
- Example use
  - Connection to keyboard/mouse
  - Serial port on PC
- Specified by *EIA*
- Voltage is +15 or -15
- Cable limited to ~50 feet
- Newer EIA standard is RS-422 (ITU standard is V.24)
- Uses *asynchronous* communication

NOTES

Asynchronous Communication

- Sender and receiver must agree on
  - Number of bits per character
  - Duration of each bit
- Receiver
  - Does not know when a character will arrive
  - May wait forever
- To ensure meaningful exchange send
  - Start bit before character
  - One or more stop bits after character

Illustration Of RS-232



- Start bit
  - Same as *0*
  - Not part of data
- Stop bit
  - Same as *1*
  - Follows data

Duration Of A Bit In RS-232C

- Determined by *baud rate*
  - Example baud rates: 9.6 Kbaud, 28.8 Kbaud, 33.6 Kbaud
  - Duration of bit is 1/baud_rate
- Sender and receiver must agree *a priori*
- Receiver samples signal
- Disagreement results in *framing error*

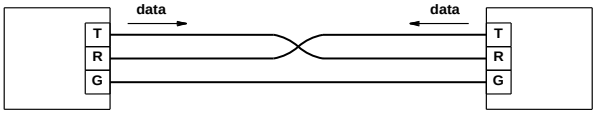
NOTES

Two-Way Communication

- Desirable in practice
- Requires each side to have transmitter and receiver
- Called *full duplex*

NOTES

Illustration Of Full-Duplex Communication

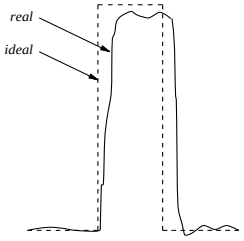


- Transmitter on one side connected to receiver on other
- Separate wires needed to carry current in each direction
- Common *ground wire*
- DB-9, DB-15, or DB-25 connector used
  - Pin *2* is transmit
  - Pin *3* is receive

Electrical Transmission (The Bad News)

- It's an ugly world
  - Electrical energy dissipates as it travels along
  - Wires have resistance, capacitance, and inductance which distort signals
  - Magnetic or electrical interference distorts signals
  - Distortion can result in loss or misinterpretation

Illustration Of Distorted Signal For A Single Bit



- In practice
  - Distortion can be much worse than illustrated

NOTES

Consequences

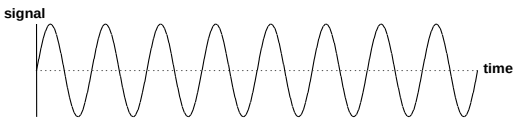
- RS-232 hardware must handle minor distortions
  - Take multiple samples per bit
  - Tolerate less than full voltage
- Cannot use electrical current for long-distance transmission

NOTES

Long-Distance Communication

- Important fact: an oscillating signal travels farther than direct current
- For long-distance communication
  - Send a sine wave (called a *carrier wave*)
  - Change (*modulate*) the carrier to encode data
- Note: modulated carrier technique used for radio and television

Illustration Of A Carrier



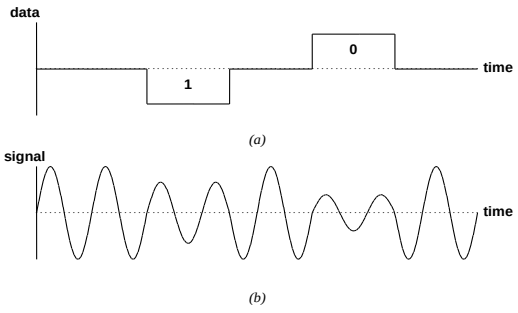
- Carrier
  - Usually a sine wave
  - Oscillates continuously
- Frequency of carrier fixed

NOTES

Types Of Modulation

- Amplitude modulation (used in AM radio)
- Frequency modulation (used in FM radio)
- Phase shift modulation (used for data)

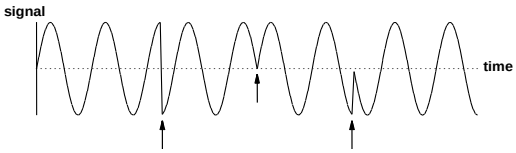
Illustration Of Amplitude Modulation



- Strength of signal encodes *0* or *1*
- One cycle of wave needed for each bit
- Data rate limited by carrier bandwidth

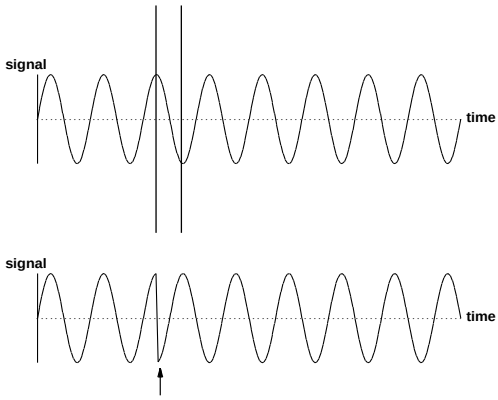
NOTES

Illustration Of Phase-Shift Modulation



- Change in phase encodes *K* bits
- Data rate higher than carrier bandwidth

Phase-Shift Example



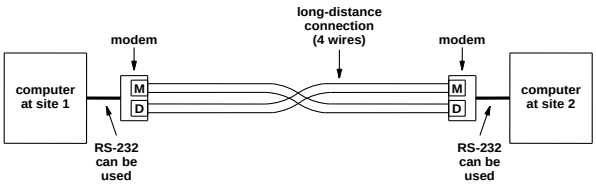
- Section of wave is omitted at phase shift
- Data bits determine size of omitted section

Modem

- Hardware device
- Used for long-distance communication
- Contains separate circuitry for
  - Modulation of outgoing signal
  - Demodulation of incoming signal
- Name abbreviates *mo*dulator/*dem*odulator

NOTES

Illustration Of Modems Used Over A Long Distance



- One modem at each end
- Separate wires carry signals in each direction
- Modulator on one modem connects to demodulator on other

Types Of Modems

- Conventional
  - Use four wires
  - Transmit modulated electrical wave
- Optical
  - Use glass fibers
  - Transmit modulated light
- Wireless
  - Use air/space
  - Transmit modulated RF wave

NOTES

Types Of Modems
(continued)

- Dialup
  - Use voice telephone system
  - Transmit modulated audio tone

- Note: in practice, a dialup modem uses multiple tones simultaneously

Illustration Of Dialup Modem

- Modem can
  - Dial
  - Answer
- Carrier is audio tone

NOTES

Modem Terminology

- *Full-duplex modem*
  - Provides 2-way communication
  - Allows simultaneous transmission
  - Uses four wires
- *Half-duplex modem*
  - Provides 2-way communication
  - Transmits in one direction at any time
  - Uses two wires

Recall

- *Propagation delay*
  - Determined by physics
  - Time required for signal to travel across medium
- *Bandwidth*
  - Electrical property of physical transmission system
  - Maximum times per second signal can change

NOTES

Fundamental Measures Of A Digital Transmission System

- *Delay*
  - The amount of time required for a bit of data to travel from one end to the other
  - Usually the same as the propagation delay in underlying hardware
- *Throughput*
  - The number of bits per second that can be transmitted
  - Related to underlying hardware bandwidth

Relationship Between Digital Throughput And Bandwidth

- Given by Nyquist's theorem:

$$D = 2B\log_2 K$$

where

  - D is maximum data rate
  - B is hardware bandwidth
  - K is number of values used to encode data

NOTES

Applications Of Nyquist's Theorem

- For RS-232
  - K is 2 because RS-232 uses two values, +15 or -15 volts, to encode data bits
  - D is $2B\log_2 2 = 2B$
- For phase-shift encoding
  - Suppose K is 8 (possible shifts)
  - D is $2B\log_2 8 = 2B \times 3 = 6B$

More Bad News

- Physics tells us that real systems emit and absorb energy (e.g., thermal)
- Engineers call unwanted energy *noise*
- Nyquist's theorem
  - Assumes a noise-free system
  - Only works in theory
- Shannon's theorem corrects for noise

NOTES

Shannon's Theorem

- Gives capacity in presence of noise:

$$C = B\log_2(1 + S/N)$$

  where

  - C is the effective channel capacity in bits per second
  - B is hardware bandwidth
  - S is the average power (*signal*)
  - N is the *noise*

- S/N is *signal-to-noise ratio*

Application Of Shannon's Theorem

- Conventional telephone system
  - Engineered for voice
  - Bandwidth is *3000 Hz*
  - Signal-to-noise ratio is approximately *1000*
  - Effective capacity is

$$3000 \log_2(1 + 1000) = \sim 30000 \text{ bps}$$

- Conclusion: dialup modems have little hope of exceeding *28.8* Kbps

NOTES

The Bottom Line

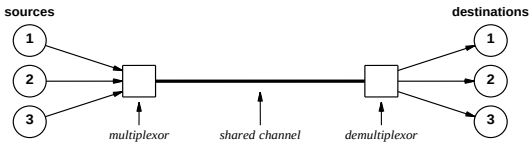
- Nyquist's theorem means finding a way to encode more bits per cycle improves the data rate
- Shannon's theorem means that no amount of clever engineering can overcome the fundamental physical limits of a real transmission system

NOTES

Multiplexing

- Fundamental to networking
- General concept
- Used in
  - Lowest level of transmission systems
  - Higher levels of network hardware
  - Protocol software
  - Applications

The General Concept Of Multiplexing



- Separate pairs of communications travel across shared channel
- Multiplexing prevents interference
- Each destination receives only data sent by corresponding source

Multiplexing Terminology

- *Multiplexor*
  - Device or mechanism
  - Accepts data from multiple sources
  - Sends data across shared channel
- *Demultiplexor*
  - Device or mechanism
  - Extracts data from shared channel
  - Sends to correct destination

NOTES

Two Basic Types Of Multiplexing

- *Time Division Multiplexing (TDM)*
  - Only one item at a time on shared channel
  - Item marked to identify source
  - Demultiplexor uses identifying mark to know where to deliver
- *Frequency Division Multiplexing (FDM)*
  - Multiple items transmitted simultaneously
  - Uses multiple ‘‘channels’’

Transmission Schemes

- *Baseband transmission*
  - Uses only low frequencies
  - Encodes data directly
- *Broadband transmission*
  - Uses multiple carriers
  - Can use higher frequencies
  - Achieves higher throughput
  - Hardware more complex and expensive

NOTES

Scientific Principle Behind Frequency Division Multiplexing

Two or more signals that use different carrier frequencies can be transmitted over a single medium simultaneously without interference.

Note: this is the same principle that allows a cable TV company to send multiple television signals across a single cable.

Wave Division Multiplexing

- Facts
  - FDM can be used with any electromagnetic radiation
  - Light is electromagnetic radiation
- When applied to light, FDM is called *wave division multiplexing*
  - Informally called *color division multiplexing*

NOTES

Summary

- Various transmission schemes and media available
  - Electrical current over copper
  - Light over glass
  - Electromagnetic waves
- Digital encoding used for data
- Asynchronous communication
  - Used for keyboards and serial ports
  - RS-232 is standard
  - Sender and receiver agree on baud rate

Summary (continued)

- Modems
  - Used for long-distance communication
  - Available for copper, optical fiber, dialup
  - Transmit modulated carrier
    - Phase-shift modulation popular
  - Classified as full- or half- duplex
- Two measures of digital communication system
  - Delay
  - Throughput

NOTES

Summary (continued)

- Nyquist's theorem
  - Relates throughput to bandwidth
  - Encourages engineers to use complex encoding
- Shannon's theorem
  - Adjusts for noise
  - Specifies limits on real transmission systems

Summary (continued)

- Multiplexing
  - Fundamental concept
  - Used at many levels
  - Applied in both hardware and software
  - Two basic types
    * Time-division multiplexing (TDM)
    * Frequency-division multiplexing (FDM)
- When applied to light, FDM is called wave-division multiplexing

NOTES

PART III

Packets, Frames, Parity, Checksums, and CRCs

NOTES

The Problem

- Cannot afford individual network connection per pair of computers
- Reasons
  - Installing wires consumes time and money
  - Maintaining wires consumes money (esp. long-distance connections)

Solution

- Network has
  - Shared central core
  - Many attached stations

The Problem With Sharing

- Demand high
- Some applications have large transfers
- Some applications cannot wait
- Need mechanism for fairness

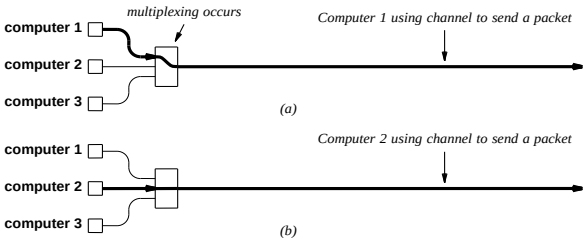
NOTES

Packet Switching Principle

- Solution for fairness
  - Divide data into small units called *packets*
  - Allow each station opportunity to send a packet before any station sends another
- Form of time-division multiplexing

NOTES

Illustration Of Packet Switching



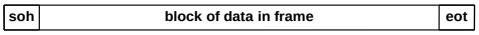
- Acquire shared medium
- Send one packet
- Allow other stations opportunity to send before sending again

Packet Details

- Depend on underlying network
  - Minimum/maximum size
  - Format
- Hardware packet called a *frame*

NOTES

Example Frame Format Used With RS-232



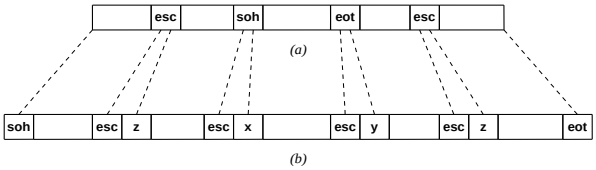
- RS-232 is character-oriented
- Special characters
  - Start of header (*soh*)
  - End of text (*eot*)

When Data Contains Special Characters

- Translate to alternative form
- Called *byte stuffing*
- Example

Character In Data	Characters Sent
soh	esc x
eot	esc y
esc	esc z

Illustration Of Frame With Byte Stuffing



- Stuffed frame longer than original
- Necessary evil

NOTES

Handling Errors

- Data can be corrupted during transmission
  - Bits lost
  - Bit values changed
- Frame includes additional information to detect/correct error
  - Set by sender
  - Checked by receiver
- Statistical guarantee

Error Detection And Recovery Techniques

- Parity bit
  - One additional bit per character
  - Can use
    * *Even parity*
    * *Odd parity*
  - Cannot handle error that changes two bits

NOTES

Error Detection And Recovery Techniques (continued)

- Checksum
  - Treat data as sequence of integers
  - Compute and send arithmetic sum
  - Handles multiple bit errors
  - Cannot handle all errors

Error Detection And Recovery Techniques (continued)

- Cyclic Redundancy Check (*CRC*)
  - Mathematical function for data
  - More complex to compute
  - Handles more errors

NOTES

Example Checksum Computation

H	e	l	l	o		w	o	r	l	d	.
48	65	6C	6C	6F	20	77	6F	72	6C	64	2E

4865 + 6C6C + 6F20 + 776F + 726C + 642E + carry = 71FC

- Checksum computed over data
- Checksum appended to frame

NOTES

Illustration Of Errors A Checksum Fails To Detect

Data Item In Binary	Checksum Value	Data Item In Binary	Checksum Value
0001	1	0011	3
0010	2	0000	0
0011	3	0001	1
0001	1	0011	3
totals	7		7

- Second bit reversed in each item
- Checksum is the same

Building Blocks For CRC

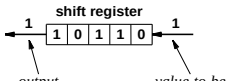
- Exclusive or

(a)

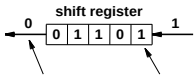
a	b	out
0	0	0
0	1	1
1	0	1
1	1	0

(b)

- Shift register



(a)

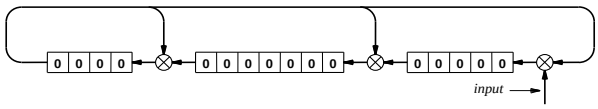


(b)

  - *a* shows status before shift
  - *b* shows status after shift
  - Output same as top bit

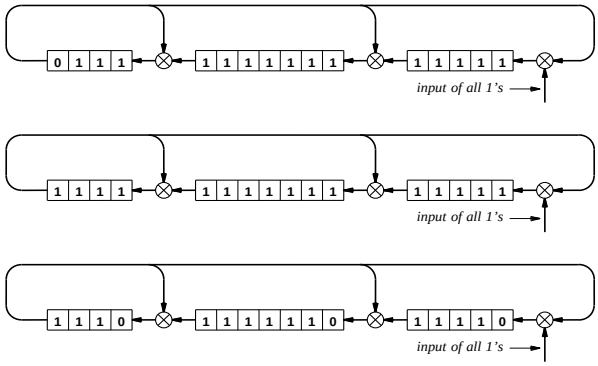
NOTES

Example Of CRC Hardware



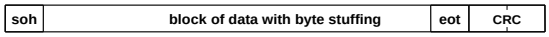
- Computes 16-bit CRC
- Registers initialized to zero
- Bits of message shifted in
- CRC found in registers

Example CRC Computation



- Input data is all *1* bits
- CRC shown after 15, 16, and 17 bits shifted
- Feedback introduces zeroes in CRC

Illustration Of Frame Using CRC



- CRC covers data only

NOTES

Summary

- Packet technology
  - Invented to provide fair access in shared network
  - Sender divides data into small packets
- Hardware packets called frames
- Can use packet-switching with RS-232
  - Special characters delimit beginning and end of frame
  - Byte-stuffing needed when special characters appear in data

Summary (continued)

- To detect data corruption
  - Sender adds information to packet
  - Receiver checks
- Techniques
  - Parity bit
  - Checksum
  - Cyclic Redundancy Check (CRC)
  - Provide statistical guarantees

NOTES

PART IV

Local Area Networks (LANs)

Classification Terminology

- Network technologies classified into three broad categories
- *Local Area Network (LAN)*
- *Metropolitan Area Network (MAN)*
- *Wide Area Network (WAN)*
- LAN and WAN most widely deployed

NOTES

The Local Area Network (LAN)

- Engineering classification
- Extremely popular (most networks are LANs)
- Many LAN technologies exist

Key Features Of A LAN

- High throughput
- Relatively low cost
- Limited to short distance
- Often rely on *shared media*

NOTES

Scientific Justification For Local Area Networks

A computer is more likely to communicate with computers that are nearby than with computers that are distant.

- Known as the *locality principle*

Topology

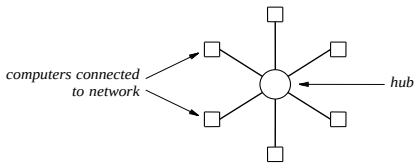
- Mathematical term
- Roughly interpreted as ‘‘geometry for curved surfaces’’

NOTES

Network Topology

- Specifies general ‘‘shape’’ of a network
- Handful of broad categories
- Often applied to LAN
- Primarily refers to interconnections
- Hides details of actual devices

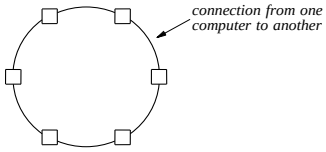
Star Topology



- Central component of network known as *hub*
- Each computer has separate connection to hub

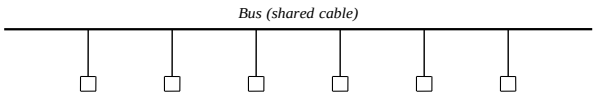
NOTES

Ring Topology



- No central facility
- Connections go directly from one computer to another

Bus Topology



- Shared medium forms main interconnect
- Each computer has a connection to the medium

NOTES

Example Bus Network: Ethernet

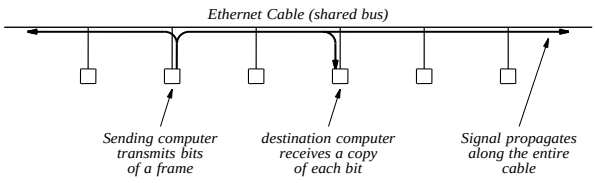
- Most popular LAN
- Widely used
- IEEE standard *802.3*
- Several generations
  - Same frame format
  - Different data rates
  - Different wiring schemes

NOTES

Shared Medium In A LAN

- Shared medium used for all transmissions
- Only one station transmits at any time
- Stations ‘‘take turns’’ using medium
- Media Access Control (*MAC*) policy ensures fairness

Illustration Of Ethernet Transmission



- Only one station transmits at any time
- Signal propagates across entire cable
- All stations receive transmission
- CSMA/CD media access scheme

CSMA/CD Paradigm

- Multiple Access (*MA*)
  - Multiple computers attach to shared media
  - Each uses same access algorithm
- Carrier Sense (*CS*)
  - Wait until medium idle
  - Begin to transmit frame
- Simultaneous transmission possible

NOTES

CSMA/CD Paradigm
(continued)

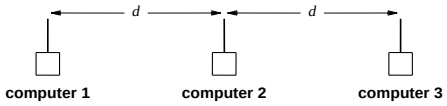
- Two simultaneous transmissions
  - Interfere with one another
  - Called *collision*
- CSMA plus *Collision Detection* (*CD*)
  - Listen to medium during transmission
  - Detect whether another station's signal interferes
  - Back off from interference and try again

Backoff After Collision

- When collision occurs
  - Wait random time t_1, $0 \le t_1 \le d$
  - Use CSMA and try again
- If second collision occurs
  - Wait random time t_2, $0 \le t_2 \le 2d$
- Double range for each successive collision
- Called *exponential backoff*

NOTES

Media Access On A Wireless Net



- Limited range
  - Not all stations receive all transmissions
  - Cannot use CSMA/CD
- Example in diagram
  - Maximum transmission distance is *d*
  - Stations *1* and *3* do not receive each other's transmissions

CSMA/CA

- Used on wireless LANs
- Both sides send small message followed by data transmission
  - ''*X* is about to send to *Y*''
  - ''*Y* is about to receive from *X*''
  - Data frame sent from *X* to *Y*
- Purpose: inform all stations in range of *X* or *Y* before transmission
- Known as *Collision Avoidance* (*CA*)

NOTES

Wi-Fi Wireless LAN Technology

- Popular
- Uses CSMA/CA for media access
- Standards set by IEEE
  - 802.11b (11 Mbps, shared channel)
  - 802.11a (54 Mbps, shared channel)
- Named *Wi-Fi* by consortium of vendors (to enhance popular appeal)

Identifying A Destination

- All stations on shared-media LAN receive all transmissions
- To allow sender to specify destination
  - Each station assigned unique number
  - Known as station's *address*
  - Each frame contains address of intended recipient

NOTES

Ethernet Addressing

- Standardized by IEEE
- Each station assigned unique 48-bit address
- Address assigned when network interface card *(NIC)* manufactured

Ethernet Address Recognition

- Each frame contains destination address
- All stations receive a transmission
- Station discards any frame addressed to another station
- Important: interface hardware, not software, checks address

NOTES

Possible Destinations

- Packet can be sent to:
  - Single destination (*unicast*)
  - All stations on network (*broadcast*)
  - Subset of stations (*multicast*)
- Address used to distinguish

Advantages Of Address Alternatives

- Unicast
  - Efficient for interaction between two computers
- Broadcast
  - Efficient for transmitting to all computers
- Multicast
  - Efficient for transmitting to a subset of computers

NOTES

Broadcast On Ethernet

- All *1s* address specifies broadcast
- Sender
  - Places broadcast address in frame
  - Transmits one copy on shared network
  - All stations receive copy
- Receiver always accepts frame that contains
  - Station's unicast address
  - The broadcast address

Multicast On Ethernet

- Half of addresses reserved for multicast
- Network interface card
  - Always accepts unicast and broadcast
  - Can accept zero or more multicast addresses
- Software
  - Determines multicast address to accept
  - Informs network interface card

NOTES

Promiscuous Mode

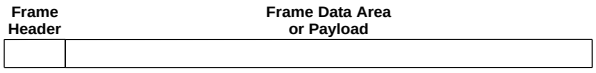
- Designed for testing/debugging
- Allows interface to accept *all* packets
- Available on most interface hardware

Identifying Frame Contents

- Integer *type field* tells recipient the type of data being carried
- Two possibilities
  - *Self-identifying* or *explicit type* (hardware records type)
  - *Implicit type* (application sending data must handle type)

NOTES

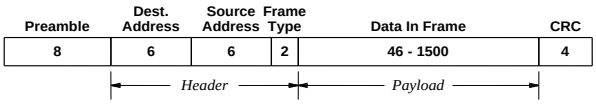
Conceptual Frame Format



- *Header*
  - Contains address and type information
  - Layout fixed
- *Payload*
  - Contains data being sent

NOTES

Illustration Of Ethernet Frame



- Sender places
  - Sender's address in *source*
  - Recipient's address in *destination*
  - Type of data in *frame type*
  - Cyclic redundancy check in *CRC*

Example Ethernet Types

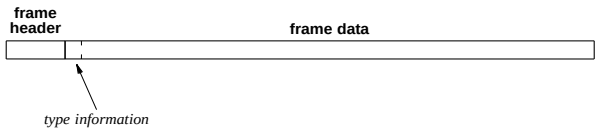
Value	Meaning
0000-05DC	Reserved for use with IEEE LLC/SNAP
0800	Internet IP Version 4
0805	CCITT X.25
0900	Ungermann-Bass Corporation network debugger
0BAD	Banyan Systems Corporation VINES
1000-100F	Berkeley UNIX Trailer encapsulation
6004	Digital Equipment Corporation LAT
6559	Frame Relay
8005	Hewlett Packard Corporation network probe
8008	AT&T Corporation
8014	Silicon Graphics Corporation network games
8035	Internet Reverse ARP
8038	Digital Equipment Corporation LANBridge
805C	Stanford University V Kernel
809B	Apple Computer Corporation AppleTalk
80C4-80C5	Banyan Systems Corporation
80D5	IBM Corporation SNA
80FF-8103	Wellfleet Communications
8137-8138	Novell Corporation IPX
818D	Motorola Corporation
FFFF	Reserved

NOTES

When Network Hardware Does Not Include Types

- Sending and receiving computers must agree
  - To only send one type of data
  - To put type information in first few octets of payload
- Most systems need type information

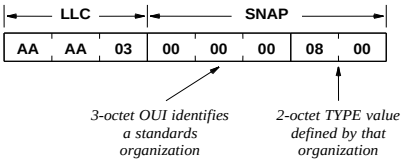
Illustration Of Type Information Added To Data



- In practice
  - Type information small compared to data carried
  - Format of type information standardized

NOTES

A Standard For Type Information



- Defined by IEEE
- Used when hardware does not include type field
- Called *LLC/SNAP header*

Demultiplexing On Type

- Network interface hardware
  - Receives copy of each transmitted frame
  - Examines address and either discards or accepts
  - Passes accepted frame to system software
- Network device software
  - Examines frame type
  - Passes frame to correct software module

Network Analyzer

- Device used for testing and maintenance
- Listens in promiscuous mode
- Produces
  - Summaries (e.g., % of broadcast frames)
  - Specific items (e.g., frames from a given address)

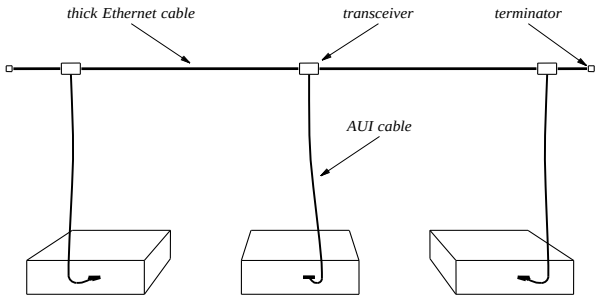
NOTES

Ethernet Wiring

- Three schemes
  - Correspond to three generations
  - All use same frame format

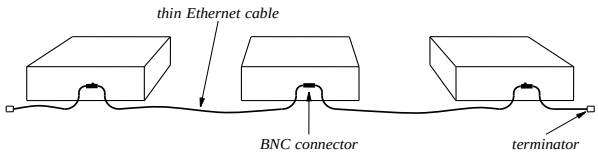
NOTES

Original Ethernet Wiring



- Used heavy coaxial cable
- Formal name *10Base5*
- Called *thicknet*

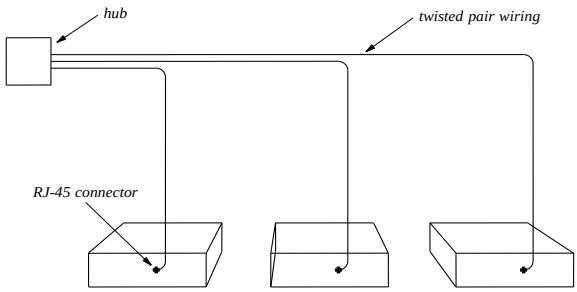
Second Generation Ethernet Wiring



- Used thinner coaxial cable
- Formal name *10Base2*
- Called *thinnet*

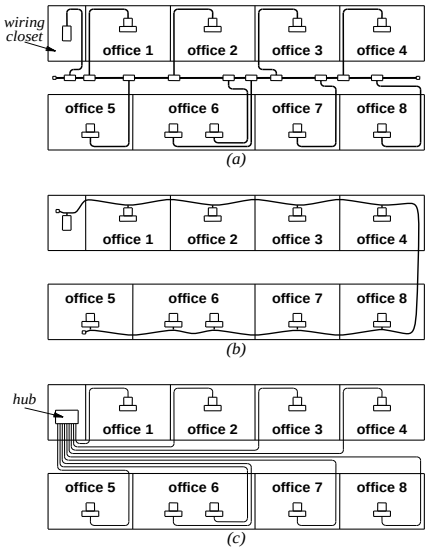
NOTES

Modern Ethernet Wiring



- Uses a *hub*
- Formal name *10Base-T*
- Called *twisted pair Ethernet*

Ethernet Wiring In An Office



A Note About Ethernet Topology

- Apparently
  - Original Ethernet used bus topology
  - Modern Ethernet uses star topology
- In fact, modern Ethernet is
  - Physical star
  - Logical bus
  - Called *star-shaped bus*

NOTES

Higher Speed Ethernets

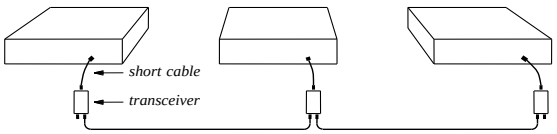
- *Fast Ethernet*
  - Operates at 100 Mbps
  - Formally *100Base-T*
  - Two wiring standards
  - 10/100 Ethernet devices available
- *Gigabit Ethernet*
  - Operates at 1000 Mbps (1 Gbps)
  - Slightly more expensive

Another LAN Using Bus Topology

- LocalTalk
  - Developed by Apple Corp.
  - Simple to use
  - Slow by current standards

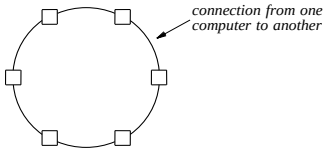
NOTES

Illustration Of LocalTalk



- Transceiver required per station
- Transceiver terminates cable

Ring Topology



- Once a popular topology for LANs
- Bits flow in single direction

NOTES

Token Passing

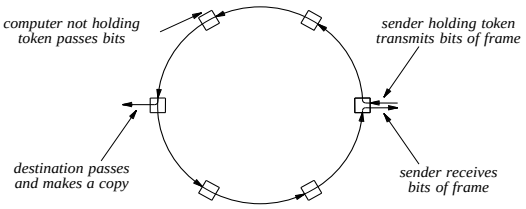
- Designed for ring topology
- Guarantees fair access
- *Token*
  - Special (reserved) message
  - Small (a few bits)

NOTES

Token Passing Paradigm

- Station
  - Waits for token to arrive
  - Transmits one packet around ring
  - Transmits token around ring
- When no station has data to send
  - Token circulates continuously

Token Passing Ring Transmission



- Station waits for token before sending
- Signal travels around entire ring
- Sender receives its own transmission

Strengths Of Token Ring Approach

- Easy detection of
  - Broken ring
  - Hardware failures
  - Interference

NOTES

Weaknesses Of Token Ring Approach

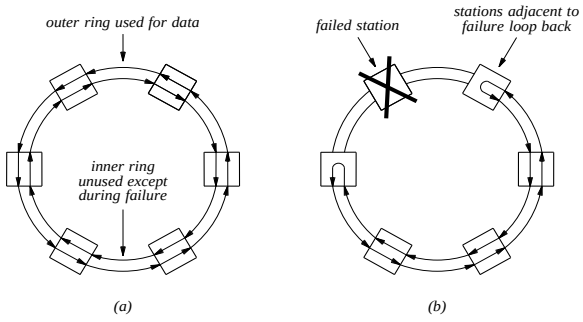
- Broken wire disables entire ring
- Point-to-point wiring
  - Awkward in office environment
  - Difficult to add/move stations

Failure Recovery In Ring Networks

- Automatic failure recovery
- Introduced by FDDI
- Uses two rings
- Terminology
  - *Dual-attached*
  - *Counter rotating*
  - *Self healing*

NOTES

Illustration Of Failure Recovery



- Normal operation uses one of two rings
- Second ring used for loopback during failure

Token Passing Ring Technologies

- *ProNet-10*
  - Operated at 10 Mbps
- *IBM Token Ring*
  - Originally operated at 4 Mbps
  - Later version operated at 16 Mbps
- *Fiber Distributed Data Interconnect (FDDI)*
  - Operates at 100 Mbps
- All are now virtually obsolete

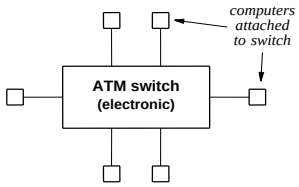
NOTES

Example Of A Physical Star Topology

- *Asynchronous Transfer Mode (ATM)*
- Designed by telephone companies
- Intended to accommodate
  - Voice
  - Video
  - Data

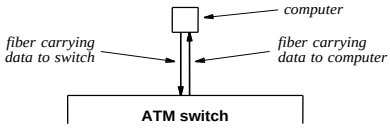
NOTES

ATM



- Building block known as *ATM switch*
- Each station connects to switch
- Switches can be interconnected

Details Of ATM Connection



- Full-duplex connections
- Two fibers used

ATM Characteristics

- High data rates (e.g. 155 Mbps)
- Fixed size packets
  - Called *cells*
  - Important for voice
- Cell size is 53 octets
  - 48 octets of data
  - 5 octets of header

NOTES

Summary

- Local Area Networks
  - Designed for short distance
  - Use shared media
  - Many technologies exist
- Topology refers to general shape
  - Bus
  - Ring
  - Star

Summary (continued)

- Address
  - Unique number assigned to station
  - Put in frame header
  - Recognized by hardware
- Address forms
  - Unicast
  - Broadcast
  - Multicast

NOTES

Summary (continued)

- Type information
  - Describes data in frame
  - Set by sender
  - Examined by receiver
- Frame format
  - Header contains address and type information
  - Payload contains data being sent

Summary (continued)

- Currently popular LAN technology
  - Ethernet (bus)
- Older LAN technologies
  - IBM Token Ring
  - FDDI (ring)
  - ATM (star)

NOTES

Summary (continued)

- Wiring and topology
  - Can distinguish
    - Logical topology
    - Physical topology (wiring)
  - Hub allows
    - Star-shaped bus
    - Star-shaped ring

NOTES

PART V

Extending Networks (Repeaters, Bridges, Switches)

Motivation

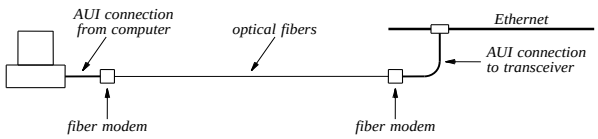
- Recall
  - Each LAN technology has a distance limitation
  - Example: CSMA/CD cannot work across arbitrary distance
- However
  - Users desire arbitrary distance connections
  - Example: two computers across a corporate campus are part of one workgroup

Extension Techniques

- Must not violate design assumptions
- Often part of original design
- Example technique
  - Use connection with lower delay than copper

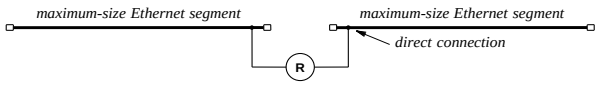
NOTES

Illustration Of Extension For One Computer



- Optical fiber
  - Has low delay
  - Has high bandwidth
  - Can pass signals within specified bounds

Repeater



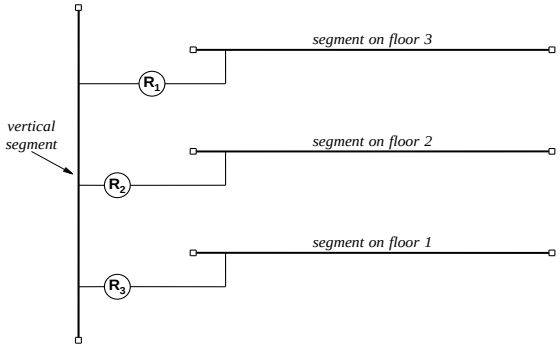
- Hardware device
- Connects two LAN segments
- Copies signal from one segment to the other
- Connection can be extended with Fiber Optic Intra-Repeater Link

NOTES

Repeater
(continued)

- Amplifies signals from one segment and sends to the other
- Operates in two directions simultaneously
- Propagates noise and collisions

Repeaters And The Original Ethernet Wiring Scheme



- Designed for office
- Only two repeaters between any pair of stations

NOTES

Hub

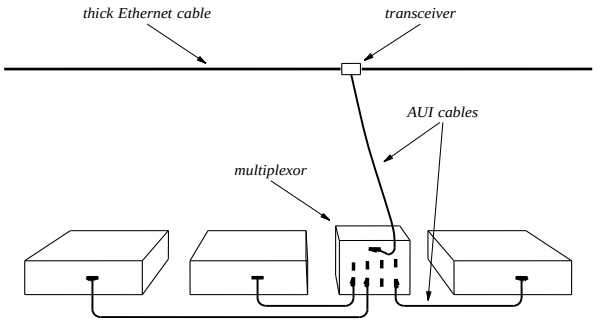
- Physically
  - Small electronic device
  - Has connections for several computers (e.g., 4 or 20)
- Logically
  - Operates on signals
  - Propagates each incoming signal to all connections
  - Similar to connecting segments with repeaters
  - Does not understand packets
- Extremely low cost

Connection Multiplexing

- Concept
  - Multiple stations share one network connection
- Motivation
  - Cost
  - Convenience of wiring
- Hardware device required

NOTES

Illustration Of Connection Multiplexing



- Multiplexing device attached to network
- Stations attach to device
- Predates hubs

Modern Equivalent Of Connection Multiplexing

- Hubs used now
- Connections on a hub
  - One for each attached computer
  - One for another hub
- Multiple hubs
  - Can be interconnected in a *daisy chain*
  - Operate as one giant hub
  - Called *stacking*

NOTES

Bridge

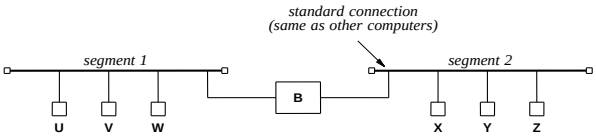
- Hardware device
- Connects two LAN segments
- Forwards frames
- Does not forward noise or collisions
- Learns addresses and filters
- Allows independent transmission

NOTES

Bridge Algorithm

- Listen in promiscuous mode
- Watch source address in incoming frames
- Make list of computers on each segment
- Only forward if necessary
- Always forward broadcast/multicast

Illustration Of A Bridge

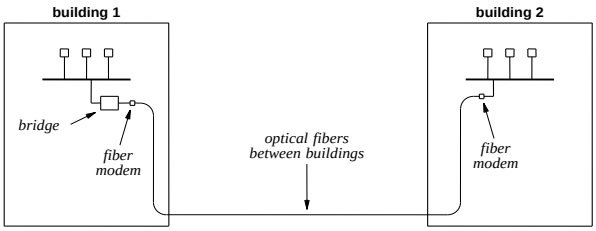


Event	Segment 1 List	Segment 2 List
Bridge boots	–	–
U sends to V	U	–
V sends to U	U, V	–
Z broadcasts	U, V	Z
Y sends to V	U, V	Z, Y
Y sends to X	U, V	Z, Y
X sends to W	U, V	Z, Y, X
W sends to Z	U, V, W	Z, Y, X

- Bridge uses source address to learn location of computers
- Learning is completely automated

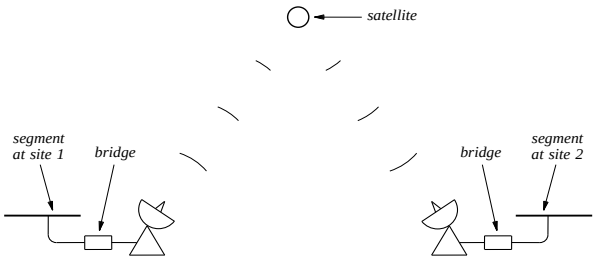
NOTES

Extending A Bridge



- Typically optical fiber
- Can span buildings

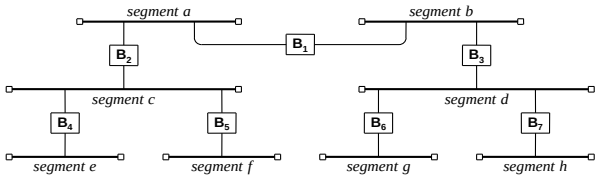
Satellite Bridging



- Can span arbitrary distance

NOTES

Apparent Problem



- Complex bridge connections may not be apparent
- Adding one more bridge inadvertently introduces a cycle
- Consider a broadcast frame

Spanning Tree Algorithm

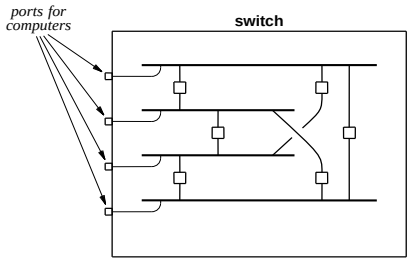
- Allows cycles
- Used by all bridges to
  - Discover one another
  - Break cycle(s)
- Known as *Distributed Spanning Tree (DST)*

Switch

- Electronic device
- Physically similar to a hub
- Logically similar to a bridge
  - Operates on packets
  - Understands addresses
  - Only forwards when necessary
- Permits separate pairs of computers to communicate at the same time
- Higher cost than hub

NOTES

Conceptual Switch Function



- Conceptual operation
  - One LAN segment per host
  - Bridge interconnects each pair of segments
- NOT an actual implementation

Summary

- LANs
  - Have distance limitations
  - Can be extended
- Fiber can be used between computer and LAN
- Repeater
  - Connects two LAN segments
  - Repeats and amplifies all signals
  - Forwards noise and collisions

NOTES

Summary
(continued)

- Bridge
  - Connects two LAN segments
  - Understands frames
  - Uses addresses
  - Does not forward noise or collisions
  - Allows simultaneous transmission on segments

Summary
(continued)

- Hub
  - Central facility in star-shaped network
  - Operates like a repeater
- Switch
  - Central facility in star-shaped network
  - Operates like a set of bridged segments

NOTES

PART VI

Long-Distance and Local Loop Digital Connection Technologies

Motivation

- Connect computers across
  - Large geographic distance
  - Public right-of-way
    * Streets
    * Buildings
    * Railroads

NOTES

Long-Distance Transmission Technologies

- General solution: lease transmission facilities from telephone company
  - Point-to-point topology
  - NOT part of conventional telephone system
  - Copper, fiber, microwave, or satellite channels available
  - Customer chooses analog or digital

Equipment For Leased Connections

- Analog circuit
  - Modem required at each end
- Digital circuit
  - DSU/CSU required at each end

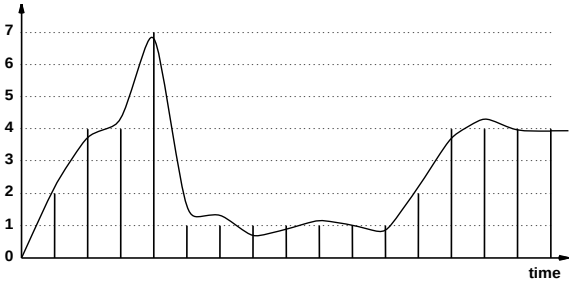
NOTES

Digital Circuit Technology

- Developed by telephone companies
- Designed for use in voice system
  - Analog audio from user's telephone converted to digital format
  - Digital format sent across network
  - Digital format converted back to analog audio

NOTES

Illustration Of Digitized Signal



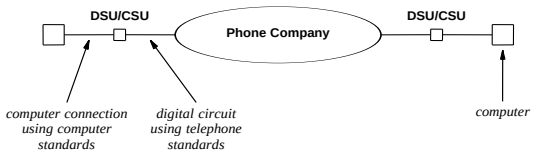
- Pick nearest digital value for each sample
- Telephone standard known as *Pulse Code Modulation (PCM)*

DSU/CSU

- Performs two functions; usually a single ‘‘box’’
- Needed because telephone industry digital encoding differs from computer industry digital encoding
- DSU portion
  - Translates between two encodings
- CSU portion
  - Terminates line
  - Allows for maintenance

NOTES

Illustration Of DSU/CSU



- Cost of digital circuit depends on
  - Distance
  - Capacity

Telephone Standards For Digital Circuits

- Specified by the telephone industry in each country
- Differ around the world
- Are known by two-character standard name
- Note: engineers refer to circuit capacity as ‘‘speed’’

NOTES

Example Circuit Capacities

Name	Bit Rate	Voice calls	Location
–	0.064 Mbps	1	
T1	1.544 Mbps	24	North America
T2	6.312 Mbps	96	North America
T3	44.736 Mbps	672	North America
E1	2.048 Mbps	30	Europe
E2	8.448 Mbps	120	Europe
E3	34.368 Mbps	480	Europe

- Note: T2 not popular

Common Digital Circuit Terminology

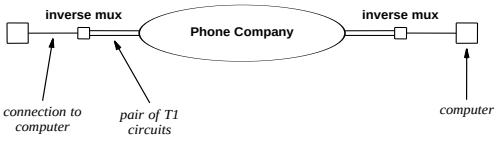
- Most common in North America
  - T1 circuit
  - T3 circuit (28 times T1)
- Also available
  - Fractional T1 (e.g., 64 Kbps circuit)

NOTES

Inverse Multiplexing

- Combines two or more circuits
- Produces intermediate capacity circuit
- Special hardware required
  - Needed at each end
  - Called *inverse multiplexor*

Example Of Inverse Multiplexing



- Can alternate between circuits for
  - Every other bit
  - Every other byte

High-Capacity Digital Circuits

- Also available from phone company
- Use optical fiber
- Electrical standards called *Synchronous Transport Signal (STS)*
- Optical standards called *Optical Carrier (OC)*

NOTES

High-Capacity Circuits

Standard Name	Optical Name	Bit Rate	Voice Calls
STS-1	OC-1	51.840 Mbps	810
STS-3	OC-3	155.520 Mbps	2430
STS-12	OC-12	622.080 Mbps	9720
STS-24	OC-24	1,244.160 Mbps	19440
STS-48	OC-48	2,488.320 Mbps	38880

- *STS-* is standard for electrical signals
- *OC-* is standard for optical signals
- Engineers usually use *OC-* terminology for everything
- *OC-3* popular

Local Loop

- Telephone terminology
- Refers to connection between residence/business and central office
- Crosses public right-of-way
- Originally for analog POTS (Plain Old Telephone Service)

NOTES

Digital Local Loop Technologies

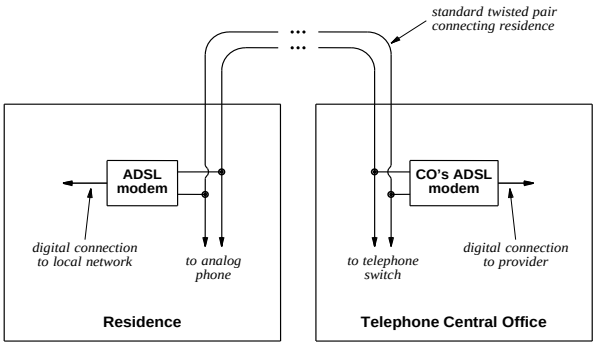
- *Integrated Services Digital Network (ISDN)*
  - Handles voice and data
  - Relatively high cost for low bandwidth
- *Digital Subscriber Line (DSL)*
- *Cable modems*
- *Hybrid Fiber Coax*

Asymmetric Digital Subscriber Line (ADSL)

- Popular DSL variant
- Runs over conventional POTS wiring
- Higher capacity downstream
- Uses frequencies above POTS

NOTES

Illustration Of ADSL Wiring



- Downstream can reach 6.4 Mbps
- Upstream can reach 640 Kbps

Cable Modems

- Send/receive over CATV wiring
- Use FDM
- Group of subscribers in neighborhood share bandwidth

NOTES

Hybrid Fiber Coax

- Wiring scheme for cable to allow digital access
- Optical fiber
  - Highest bandwidth
  - Extends from central office to neighborhood concentration points
- Coaxial cable
  - Less bandwidth
  - Extends from neighborhood concentration point to individual subscribers (e.g., residence)

Summary

- Technologies exist that span long distances
  - Leased analog lines (require modems)
  - Leased digital circuits (require DSU/CSUs)
- Digital circuits
  - Available from phone company
  - Cost depends on distance and capacity
  - Popular capacities called T1 and T3
  - Fractional T1 also available

NOTES

Summary
(continued)

- High capacity circuits available
  - Popular capacities known as OC-3, OC-12
- Local loop refers to connection between central office and subscriber
- Local loop technologies include
  - DSL (especially ADSL)
  - Cable modems

NOTES

PART VII

Wide Area Networks (WANs), Routing, and Shortest Paths

Motivation

- Connect multiple computers
- Span large geographic distance
- Cross public right-of-way
  - Streets
  - Buildings
  - Railroads

Building Blocks

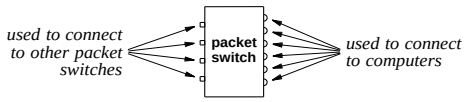
- Point-to-point long-distance connections
- *Packet switches*

NOTES

Packet Switch

- Hardware device
- Connects to
  - Other packet switches
  - Computers
- Forwards packets
- Uses addresses

Illustration Of A Packet Switch



- Special-purpose computer system
  - CPU
  - Memory
  - I/O interfaces
  - Firmware

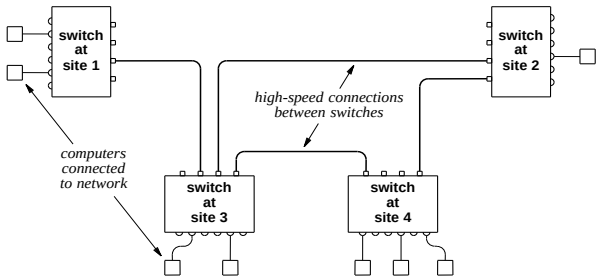
NOTES

Building A WAN

- Place one or more packet switches at each site
- Interconnect switches
  - LAN technology for local connections
  - Leased digital circuits for long-distance connections

NOTES

Illustration Of A WAN



- Interconnections depend on
  - Estimated traffic
  - Reliability needed

Store And Forward

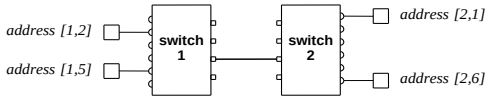
- Basic paradigm used in packet switched network
- Packet
  - Sent from source computer
  - Travels switch-to-switch
  - Delivered to destination
- Switch
  - *Stores* packet in memory
  - Examines packet's destination address
  - *Forwards* packet toward destination

Addressing In A WAN

- Need
  - Unique address for each computer
  - Efficient forwarding
- Two-part address
  - Packet switch number
  - Computer on that switch

NOTES

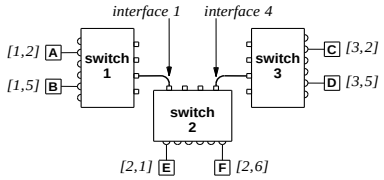
Illustration Of WAN Addressing



- Two-part address encoded as integer
  - High-order bits for switch number
  - Low-order bits for computer number

NOTES

Next-Hop Forwarding



(a)

destination	next hop
[1,2]	interface 1
[1,5]	interface 1
[3,2]	interface 4
[3,5]	interface 4
[2,1]	computer E
[2,6]	computer F

(b)

- Performed by packet switch
- Uses table of routes
- Table gives *next hop*

Forwarding Table Abbreviations

Destination	Next Hop
(1, anything)	interface 1
(3, anything)	interface 4
(2, anything)	local computer

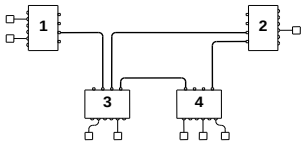
- Many entries point to same next hop
- Can be condensed (*default*)
- Improves lookup efficiency

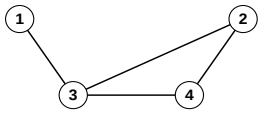
NOTES

Source Of Routing Table Information

- Manual
  - Table created by hand
  - Useful in small networks
  - Useful if routes never change
- Automatic routing
  - Software creates/updates table
  - Needed in large networks
  - Changes routes when failures occur

Relationship Of Routing To Graph Theory





destination	next hop
1	-
2	(1,3)
3	(1,3)
4	(1,3)

node 1

destination	next hop
1	(2,3)
2	-
3	(2,3)
4	(2,4)

node 2

destination	next hop
1	(3,1)
2	(3,2)
3	-
4	(3,4)

node 3

destination	next hop
1	(4,3)
2	(4,2)
3	(4,3)
4	-

node 4

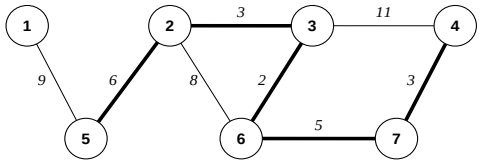
- Graph
  - *Node* models switch
  - *Edge* models connection

Shortest Path Computation

- Algorithms from graph theory
- No central authority (distributed computation)
- A switch
  - Must learn route to each destination
  - Only communicates with directly attached neighbors

NOTES

Illustration Of Minimum Weight Path



- Label on edge represents ‘‘distance’’
- Possible distance metric
  - Geographic distance
  - Economic cost
  - Inverse of capacity
- Darkened path is minimum *4* to *5*

Algorithms For Computing Shortest Paths

- *Distance Vector (DV)*
  - Switches exchange information in their routing tables
- *Link-state*
  - Switches exchange link status information
- Both used in practice

NOTES

Distance Vector

- Periodic, two-way exchange between neighbors
- During exchange, switch sends
  - List of pairs
  - Each pair gives *(destination, distance)*
- Receiver
  - Compares each item in list to local routes
  - Changes routes if better path exists

NOTES

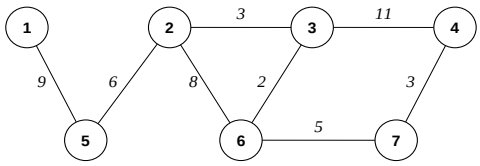
Distance Vector Algorithm

```
Given:
    a local routing table, a weight for each link that connects to another
    switch, and an incoming routing message
Compute:
    an updated routing table
Method:
    Maintain a distance field in each routing table entry;
    Initialize routing table with a single entry that has the destination equal to the local packet switch,
    the next-hop unused, and the distance set to zero;
    Repeat forever {
        wait for the next routing message to arrive over the network from a neighbor;  Let N be the sending switch;
        for each entry in the message {
            Let V be the destination in the entry and let D be the distance;
            Compute C as D plus the weight assigned to the link over which the message arrived;
            Examine and update the local routing table:
            if (no route exists to V) {
                add an entry to the local routing table for destination
                    V with next-hop N and distance C;
            } else if (a route exists that has next-hop N) {
                    replace the distance in existing route with C;
            } else if (a route exists with distance greater than C) {
                    change the next-hop to N and distance to C;
            }
    }
```


Distance Vector Intuition

- Let
  - *N* be neighbor that sent the routing message
  - *V* be destination in a pair
  - *D* be distance in a pair
  - *C* be *D* plus the cost to reach the sender
- If no local route to *V* or local route has cost greater than *C*, install a route with next hop *N* and cost *C*
- Else ignore pair

Example Of Distance Vector Routing



- Consider transmission of one DV message
- Node *2* sends to nodes *3*, *5*, and *6*
- Node *6* installs cost *8* route to node *2*
- Later, node *3* sends update
- Node *6* changes route to make node *3* the next hop for destination *2*

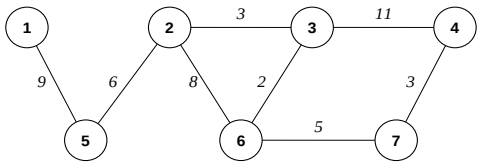
NOTES

Link-State Routing

- Overcomes instabilities in DV
- Pair of switches periodically
  - Test link between them
  - Broadcast link status message
- Switch
  - Receives status messages
  - Computes new routes
  - Uses Dijkstra's algorithm

NOTES

Example Of Link-State Information



- Assume nodes *2* and *3*
  - Test link between them
  - Broadcast information
- Each node
  - Receives information
  - Recomputes routes as needed

Dijkstra's Shortest Path Algorithm

- Input
  - Graph with weighted edges
  - Node *n*
- Output
  - Set of shortest paths from *n* to each node
  - Cost of each path
- Called *Shortest Path First (SPF)* algorithm

Dijkstra's Algorithm

```
Given:
    a graph with a nonnegative weight assigned to each edge and a designated source node
Compute:
    the shortest distance from the source node to each other node and a next-hop routing table
Method:
   Initialize set S to contain all nodes except the source node;
   Initialize array D so that D[v] is the weight of the edge from the source to v if such an edge exists, and infinity
otherwise;
   Initialize entries of R so that R[v] is assigned v if an edge exists from the source to v, and zero otherwise;

   while (set S is not empty) {
      choose a node u from S such that D[u] is minimum;
      if (D[u] is infinity) {
         no path exists to nodes in S; quit;
}
      delete u from set S;
      for each node v such that (u,v) is an edge {
         if (v is still in S) {
            c = D[u] + weight(u,v);
            if (c < D[v])  {
               R[v] = u;
               D[v] = c;
            }
         }
      }
   }
```

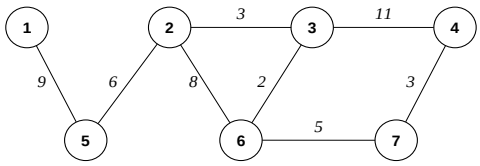

NOTES

Algorithm Intuition

- Start with self as source node
- Move outward
- At each step
  - Find node *u* such that it
    - Has not been considered
    - Is ‘‘closest’’ to source
  - Compute
    - Distance from *u* to each neighbor *v*
    - If distance shorter, make path from *u* go through *v*

NOTES

Result Of Dijkstra's Algorithm



- Example routes from node *6*
  - To *3*, next hop = *3*, cost = *2*
  - To *2*, next hop = *3*, cost = *5*
  - To *5*, next hop = *3*, cost = *11*
  - To *4*, next hop = *7*, cost = *8*

Early WAN Technologies

- ARPANET
  - Historically important in packet switching
  - Fast when invented; slow by current standards
- X.25
  - Early commercial service
  - Still used
  - More popular in Europe

Recent WAN Technologies

- SMDS
  - Offered by phone companies
  - Not as popular as Frame Relay
- Frame Relay
  - Widely used commercial service
  - Offered by phone companies
- ATM

NOTES

Asynchronous Transfer Mode (ATM)

- Designed by phone companies
- Single technology meant to handle
  - Voice
  - Video
  - Data
- Intended as LAN or WAN
- Goal: replacement for Internet

ATM Characteristics

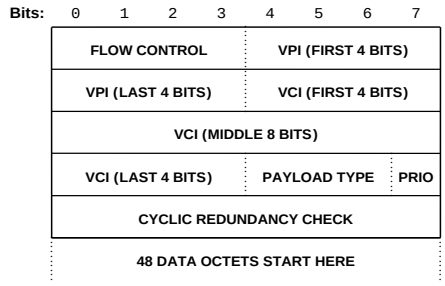
- End-to-end (application to application)
- Connection-oriented interface:
  - Establish connection
  - Send data
  - Close connection
- Performance guarantees (statistical)
- Uses cell switching

NOTES

ATM Cell

- Fixed size packet (for highest speed electronics)
- Size chosen as compromise between voice (small) and data (large)
  - 5 octet header
  - 48 octet payload
- Note: size not optimal for any application

ATM Cell Header



NOTES

ATM Switch

- Building block of ATM network
- Connections to
  - Computers
  - Other ATM switches
- Accepts and forwards cells

Cell Forwarding

- Performed directly by hardware
- Incoming cell sent to an outgoing interface
- Uses label in cell
- Motivation: highest speed

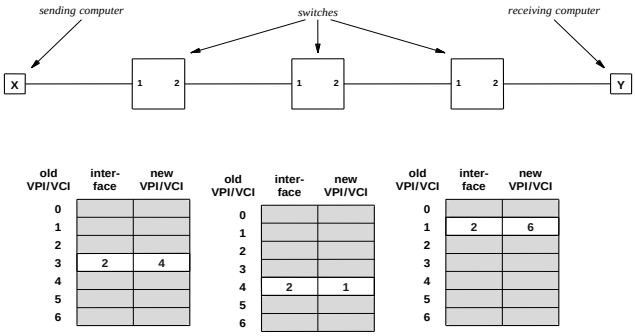
NOTES

Label Switching

- ATM connection identified with 24-bit binary value
  - Known as *Virtual Path Identifier / Virtual Channel Identifier (VPI / VCI)*
  - Generically called *label*
- VPI / VCI rewritten at each switch

NOTES

Example Of VPI/VCI Rewriting



ATM Quality Of Service

- Fine-grained (per connection)
- Specified when connection established
- Endpoint specifies
  - Type of data transfer
  - Throughput desired
  - Maximum packet burst size
  - Maximum delay tolerated

Type Of Data Transfer

- Constant Bit Rate (CBR)
  - Example: audio
- Variable Bit Rate (VBR)
  - Example: video with adaptive encoding
- Available Bit Rate (ABR)
  - Example: data
- Unspecified Bit Rate (UBR)
- Each type has detailed parameters (e.g., mean, max, burst duration)

NOTES

Sending Data Over ATM

- Uses *ATM Adaptation Layer 5 (AAL5)*
- Accepts and delivers large, variable-size packets
- AAL5 divides into cells for transmission
  - Called *segmentation and reassembly*

Assessment Of ATM

- Failed to deliver on promise
- Switches too expensive for LAN
- QoS impossible to implement

NOTES

Summary

- Wide Area Networks (WANs)
  - Span long distances
  - Connect many computers
  - Built from packet switches
  - Use store-and-forward
- WAN addressing
  - Two-part address
  - Switch/computer

Summary (continued)

- Routing
  - Each switch contains routing table
  - Table gives next-hop for destination
- Routing tables created
  - Manually
  - Automatically
- Two basic routing algorithms
  - Distance vector
  - Link state

NOTES

Summary (continued)

- Example WAN technologies
  - ARPANET
  - X.25
  - SMDS
  - Frame Relay
  - ATM

NOTES

PART VIII

Network Properties

(Ownership, Service Paradigm, Measures of Performance)

Network Ownership And Service Type

- *Private*
  - Owned by individual or corporation
  - Restricted to owner's use
  - Typically used by large corporations
- *Public*
  - Owned by a common carrier
  - Individuals or corporations can subscribe
  - ''Public'' refers to availability, not data

Advantages And Disadvantages

- Private
  - Complete control
  - Installation and operation costs
- Public
  - No need for staff to install / operate network
  - Dependency on carrier
  - Subscription fee

NOTES

Public Network Connections

- One connection per subscriber
  - Typical for small corporation or individual
  - Communicate with another subscriber
- Multiple connections per subscriber
  - Typical for large, multi-site corporation
  - Communicate among multiple sites as well as with another subscriber

Virtual Private Network

- A service
- Provided over public network
- Interconnects sites of single corporation
- Acts like private network
  - No packets sent to other subscribers
  - No packets received from other subscribers
  - Data encrypted

NOTES

Network Service Paradigm

- Fundamental characteristic of network
  - Understood by hardware
  - Visible to applications
- Two basic types of networks
  - Connectionless
  - Connection-oriented

Connectionless (CL)

- Sender
  - Forms packet to be sent
  - Places address of intended recipient in packet
  - Transfers packet to network for delivery
- Network
  - Uses destination address to forward packet
  - Delivers

NOTES

Characteristics Of Connectionless Networks

- Packet contains identification of destination
- Each packet handled independently
- No setup required before transmitting data
- No cleanup required after sending data
- Think of postcards

Connection-Oriented (CO)

- Sender
  - Requests connection to receiver
  - Waits for network to form connection
  - Leaves connection in place while sending data
  - Terminates connection when no longer needed

NOTES

Connection-Oriented (CO)
(continued)

- Network
  - Receives connection request
  - Forms path to specified destination and informs sender
  - Transfers data across connection
  - Removes connection when sender requests
- Think of telephone calls

Terminology

- In conventional telephone system
  - *Circuit*
- In CO data network
  - *Virtual Circuit*
  - *Virtual Channel*

NOTES

Comparison Of CO and CL

- CO
  - More intelligence in network
  - Can reserve bandwidth
  - Connection setup overhead
  - State in packet switches
  - Well-suited to real-time applications
- CL
  - Less overhead
  - Permits asynchronous use
  - Allows broadcast/multicast

Two Connection Types

- *Permanent Virtual Circuit (PVC)*
  - Entered manually
  - Survives reboot
  - Usually persists for months
- *Switched Virtual Circuit (SVC)*
  - Requested dynamically
  - Initiated by application
  - Terminated when application exits

NOTES

Examples Of Service Paradigm Various Technologies Use

Technology	CO	CL	used for LAN	used for WAN
Ethernet		•	•	
Wi-Fi		•	•	
Frame Relay	•			•
SMDS		•		•
ATM	•		•	•
LocalTalk		•	•	

Connection Multiplexing

- Typical computer has one physical connection to network
- All logical connections multiplexed over physical interconnection
- Data transferred must include *connection identifier*

NOTES

Connection Identifier

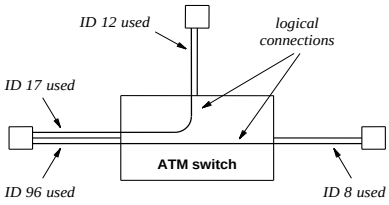
- Integer value
- One per active VC
- Not an address
- Allows multiplexing
  - Computer supplies when sending data
  - Network supplies when delivering data

NOTES

Example Connection Identifier (ATM)

- 24 bits long
- Divided into two parts
  - *Virtual Path Identifier*
  - *Virtual Channel Identifier*
- Known as *(VPI/VCI)*
- Different at each end of connection
  - Mapped by switches

Illustration Of ATM VC



- Switch maps VPI/VCIs
  - *17* to *12*
  - *96* to *8*

Two Primary Performance Measures

- Delay
- Throughput

NOTES

Delay

- Time required for one bit to travel through the network
- Three types (causes)
  - Propagation delay
  - Switching delay
  - Queuing delay
- Intuition: ‘‘length’’ of the pipe

Throughput

- Number of bits per second that can be transmitted
- Capacity
- Intuition: ‘‘width’’ of the pipe

NOTES

Components Of Delay

- Fixed (nearly constant)
  - Propagation delay
  - Switching delay
- Variable
  - Queuing delay
  - Depends on throughput

Relationship Between Delay And Throughput

- When network idle
  - Queuing delay is zero
- As load on network increases
  - Queuing delay rises
- Load defined as ratio of throughput to capacity
  - Called *utilization*

NOTES

Relationship Between Delay And Utilization

- Define
  - D_0 to be the propagation and switching delay
  - U to be the utilization $(0 \leq U \leq 1)$
  - D to be the total delay
- Then

$$D = \frac{D_0}{(1 - U)}$$

- High utilization known as *congestion*

Practical Consequence

Any network that operates with a utilization approaching 100% of capacity is doomed.

NOTES

Delay-Throughput Product

- Delay
  - Time to cross network
  - Measured in seconds
- Throughput
  - Capacity
  - Measured in bits per second
- Delay × Throughput
  - Measured in bits
  - Gives quantity of data ‘‘in transit’’

Summary

- Network can be
  - Public
  - Private
- Virtual Private Network
  - Uses public network
  - Connects set of private sites
  - Addressing and routing guarantee isolation

NOTES

Summary (continued)

- Networks are
  - Connectionless
  - Connection-Oriented
- Connection types
  - Permanent Virtual Circuit
  - Switched Virtual Circuit
- Two performance measures
  - Delay
  - Throughput

Summary (continued)

- Delay and throughput interact
- Queueing delay increases as utilization increases
- Delay × Throughput
  - Measured in bits
  - Gives total data ‘‘in transit’’

NOTES

PART IX

Protocols and Protocol Layering

Protocol

- Agreement about communication
- Specifies
  - Format of messages
  - Meaning of messages
  - Rules for exchange
  - Procedures for handling problems

NOTES

Need For Protocols

- Hardware is low level
- Many problems can occur
  - Bits corrupted or destroyed
  - Entire packet lost
  - Packet duplicated
  - Packets delivered out of order

Need For Protocols
(continued)

- Need mechanisms to distinguish among
  - Multiple computers on a network
  - Multiple applications on a computer
  - Multiple copies of a single application on a computer

NOTES

Set Of Protocols

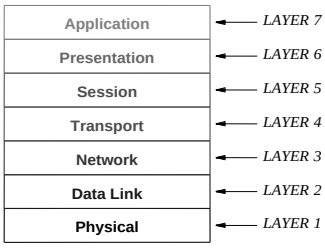
- Work together
- Each protocol solves part of communication problem
- Known as
  - *Protocol suite*
  - *Protocol family*
- Designed in *layers*

Plan For Protocol Design

- Intended for protocol designers
- Divides protocols into *layers*
- Each layer devoted to one subproblem
- Example: ISO 7-layer reference model

NOTES

Illustration Of The 7-Layer Model



- Defined early
- Now somewhat dated
- Does not include Internet layer!

ISO Layers

- Layer 1: Physical
  - Underlying hardware
- Layer 2: Data Link (media access)
  - Hardware frame definitions
- Layer 3: Network
  - Packet forwarding
- Layer 4: Transport
  - Reliability

NOTES

ISO Layers
(continued)

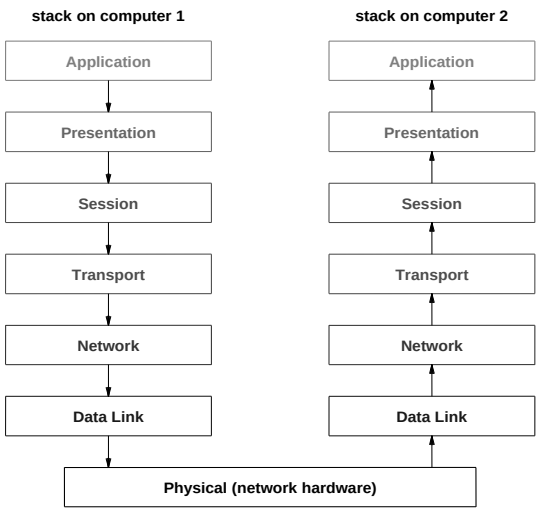
- Layer 5: Session
  - Login and passwords
- Layer 6: Presentation
  - Data representation
- Layer 7: Application
  - Individual application program

Layers And Protocol Software

- Protocol software follows layering model
  - One software module per layer
  - Modules cooperate
  - Incoming or outgoing data passes from one module to another
- Entire set of modules known as *stack*

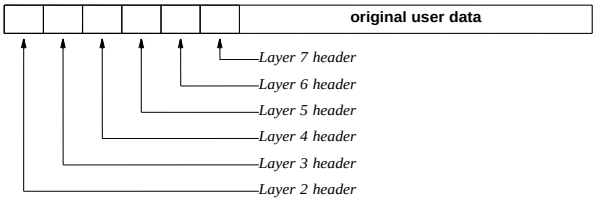
NOTES

Illustration Of Stacks



NOTES

Layers And Packet Headers



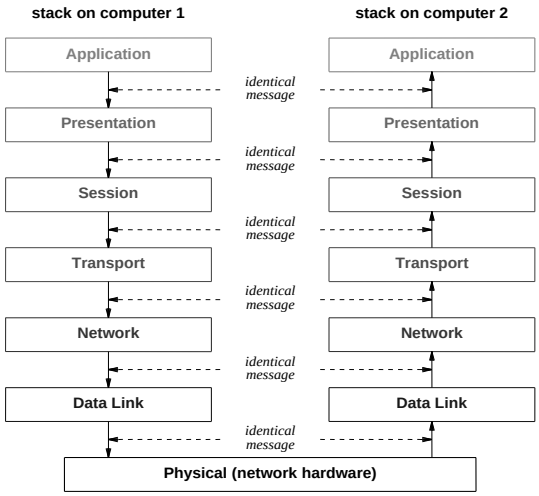
- Each layer
  - Prepends header to outgoing packet
  - Removes header from incoming packet

Scientific Layering Principle

Software implementing layer N at the destination receives exactly the message sent by software implementing layer N at the source.

NOTES

Illustration Of Layering Principle



Protocol Techniques

- For bit corruption
  - Parity
  - Checksum
  - CRC
- For out-of-order delivery
  - Sequence numbers
- Duplication
  - Sequence numbers

Protocol Techniques
(continued)

- For lost packets
  - Positive acknowledge and retransmission
- For replay (excessive delay)
  - Unique message ID
- For data overrun
  - Flow control

NOTES

Flow Control

- Needed because
  - Sending computer faster than receiving computer
  - Sending application faster than receiving application
- Related to buffering
- Two forms
  - Stop-and-go
  - Sliding window

Stop-And-Go Flow Control

- Sending side
  - Transmits one packet
  - Waits for signal from receiver
- Receiving side
  - Receives and consumes packet
  - Transmits signal to sender
- Inefficient

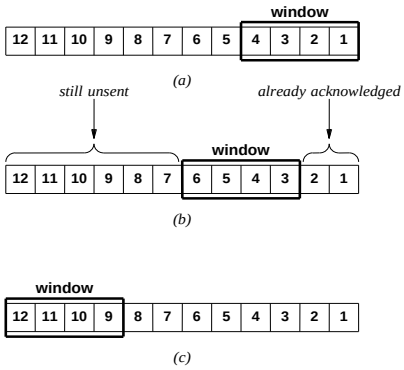
NOTES

Sliding Window Flow Control

- Receiving side
  - Establishes multiple buffers and informs sender
- Sending side
  - Transmits packets for *all* available buffers
  - Only waits if no signal arrives before transmission completes
- Receiving side
  - Sends signals as packets arrive

NOTES

Illustration Of Sliding Window On Sending Side



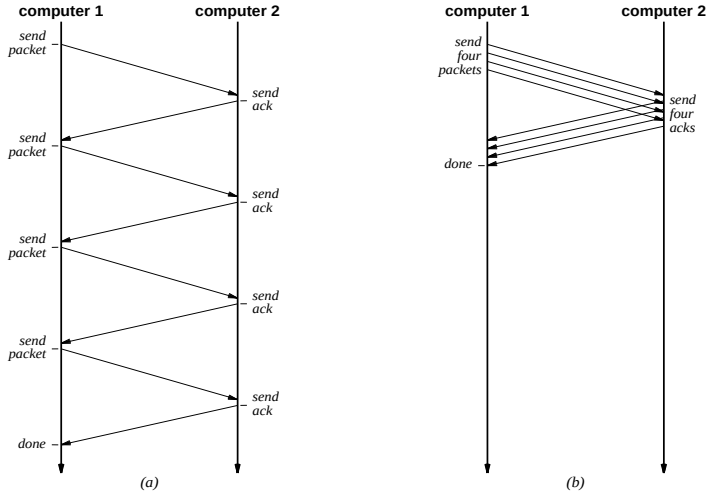
- Window tells how many packets can be sent
- Window moves as acknowledgements arrive

Performance

- Stop-and-go
  - Slow
  - Useful only in special cases
- Sliding window
  - Fast
  - Needed in high-speed network

NOTES

Comparison Of Flow Control



Why Sliding Window?

- Simultaneously
  - Increase throughput
  - Control flow
- Speedup

$$T_w = min(B, T_g \times W)$$

where

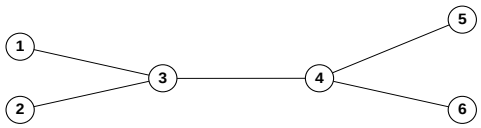
* T_w is sliding window throughput
* B is underlying hardware bandwidth
* T_g is stop-and-go throughput
* W is window size.

Congestion

- Fundamental problem in networks
- Caused by traffic, not hardware failure
- Analogous to congestion on a highway
- Principle cause of delay

NOTES

Illustration Of Architecture That Can Experience Congestion



- Multiple sources
- Bottleneck

Congestion And Loss

Modern network hardware works well; most packet loss results from congestion, not from hardware failure.

NOTES

Avoiding Congestion

- Rate control
  - Limit rate of data transmission
  - Performed by sending computer
  - Performed by network
- Network rate control
  - Monitor incoming traffic
  - Drop or reject packets over rate
  - Called *traffic shaping*

Summary

- Protocols
  - Rules for communication
  - Specify syntax and semantics
  - Complex
- Protocol layering
  - Intended for protocol designers
  - Helps organize set of protocols
  - Each layer handles one problem

NOTES

Summary
(continued)

- Problems and techniques
  - Corruption: parity, checksums, CRCs
  - Duplication, out-of-order delivery: sequence numbers
  - Loss: acknowledgement and retransmission
  - Replay: unique ID
  - Congestion: rate control
  - Data overrun: flow control

Summary (continued)

- Two types of flow control
  - Stop-and-go
  - Sliding window
- Sliding window
  - Receiver advertises buffer
  - Sender can fill entire buffer
  - Produces higher performance

NOTES

PART X

Internetworking Part 1

(Concept, IP Addressing, IP Routing, IP Datagrams, Address Resolution)

Motivation For Internetworking

- LANs
  - Low cost
  - Limited distance
- WANs
  - High cost
  - Unlimited distance

NOTES

Heterogeneity Is Inevitable

No single networking technology best for all needs.

Universal Service

- Fundamental concept in networking
- Pioneered by telephone system
- Arbitrary pair of computers can communicate
- Desirable
- Difficult in a heterogeneous world

NOTES

Heterogeneity And Universal Service

- Incompatibilities among networks
  - Electrical properties
  - Signaling and data encoding
  - Packet formats
  - Addresses

The Bottom Line

Although universal service is highly desirable, incompatibilities among network hardware and physical addressing prevent an organization from building a bridged network that includes arbitrary technologies.

NOTES

An Internetwork

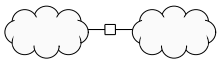
- Begin with heterogeneous network technologies
- Connect the physical networks
- Create software to make resulting system appear homogeneous
- Called an *internetwork* or *internet*

Connecting Heterogeneous Networks

- Computer system used
  - Special-purpose
  - Dedicated
  - Works with LAN or WAN technologies
  - Known as
    * *Internet router*
    * *Internet gateway*

NOTES

Illustration Of An Internet Router



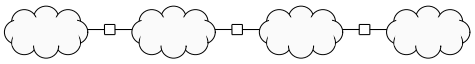
- Cloud denotes arbitrary network technology
- One interface per network

Important Idea

A router can interconnect networks that use different technologies, including different media and media access techniques, physical addressing schemes, or frame formats.

NOTES

Internet Architecture



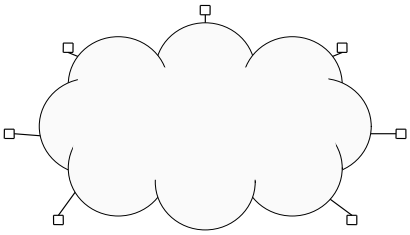
- Multiple
  - Networks
  - Routers interconnecting networks
- *Host* computer connects to a network
- Single router has insufficient
  - CPU power and memory
  - I/O capability

NOTES

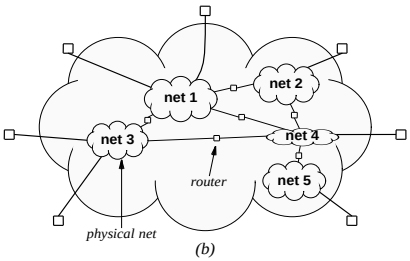
Internetworking

- Goal: communication system
  - Seamless
  - Uniform
  - General-purpose
  - Universal
  - Hides heterogeneity from user

The Internet Concept



(a) The Goal



(b)

To Hide Heterogeneity

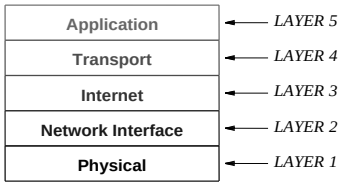
- Create ‘‘virtual’’ network
- Invent
  - Addressing scheme
  - Naming scheme
- Implement with
  - Protocol software
- Note: protocol software needed on both hosts and routers

NOTES

Internet Protocols

- Known as TCP/IP
- Many protocols comprise *suite*
- Designed to work together
- Divided into five conceptual layers

Layering Used With TCP/IP



- Note: TCP/IP layering model replaces the old ISO model

NOTES

TCP/IP Layers

- Layer 1: Physical
  - Basic network hardware
- Layer 2: Network Interface
  - MAC frame format
  - MAC addressing
  - Interface between computer and network (NIC)
- Layer 3: Internet
  - Facilities to send packets across internet composed of multiple routers

TCP/IP Layers
(continued)

- Layer 4: Transport
  - Transport from an application on one computer to application on another
- Layer 5: Application
  - Everything else

NOTES

Internet Protocol (IP)

- Only protocol at Layer 3
- Fundamental in suite
- Defines
  - Internet addressing
  - Internet packet format
  - Internet routing

IP Addressing

- Abstraction
- Independent of hardware addressing
- Used by
  - Higher-layer protocols
  - Applications

NOTES

IP Address

- Virtual
  - Only understood by software
- Used for all communication
- 32-bit integer
- Unique value for each host

IP Address Assignment

An IP address does not identify a specific computer. Instead, each IP address identifies a connection between a computer and a network. A computer with multiple network connections (e.g., a router) must be assigned one IP address for each connection.

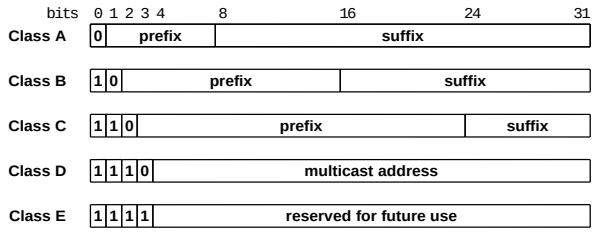
NOTES

IP Address Details

- Divided into two parts
  - Prefix identifies network
  - Suffix identifies host
- Global authority assigns unique prefix to network
- Local administrator assigns unique suffix to host

NOTES

Original Classes Of Addresses



- Initial bits determine class
- Class determines boundary between prefix and suffix

Dotted Decimal Notation

- Shorthand for IP address
- Allows humans to avoid binary
- Represents each octet in decimal separated by dots
- NOT the same as names like *www.somewhere.com*

NOTES

Example Of Dotted Decimal Notation

32-bit Binary Number	Equivalent Dotted Decimal
10000001 00110100 00000110 00000000	129 . 52 . 6 . 0
11000000 00000101 00110000 00000011	192 . 5 . 48 . 3
00001010 00000010 00000000 00100101	10 . 2 . 0 . 37
10000000 00001010 00000010 00000011	128 . 10 . 2 . 3
10000000 10000000 11111111 00000000	128 . 128 . 255 . 0

- Four decimal values per 32-bit address
- Each decimal number
  - Represents eight bits
  - Is between *0* and *255*

Classful Addresses And Network Sizes

Address Class	Prefix Bits	Max Nets	Suffix Bits	Max Hosts Per Net
A	7	128	24	16777216
B	14	16384	16	65536
C	21	2097152	8	256

- Maximum network size determined by class of address
- Class *A* large
- Class *B* medium
- Class *C* small

NOTES

Addressing Example

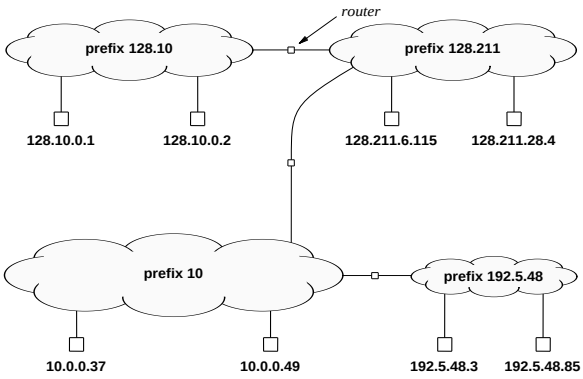
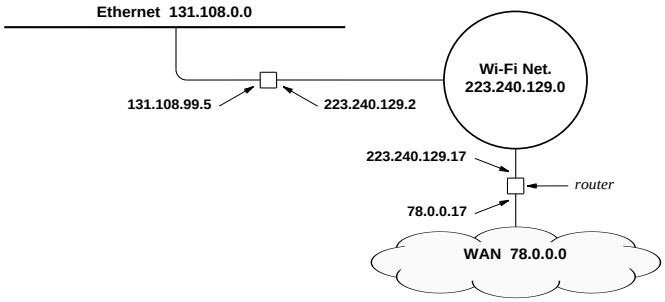


Illustration Of Router Addresses



- Address prefix identifies network
- Need one router address per connection

NOTES

Special Addresses

Prefix	Suffix	Address Type	Purpose
all-0s	all-0s	this computer	bootstrap
network	all-0s	network	network ID
network	all-1s	directed bcast	bcast on specified net
all-1s	all-1s	limited bcast	bcast on local net
127	any	loopback	testing

- Network address not used in packets
- Loopback never leaves local computer

Subnet And Classless Addressing

- Not part of original scheme
- Invented to prevent address exhaustion
- Allow boundary between prefix and suffix to occur on arbitrary bit boundary
- Require auxiliary information to identify boundary

Address Mask

- Accompanies IP address
- 32 bit binary value
- Specifies prefix / suffix boundary
  - 1 bits cover prefix
  - 0 bits cover suffix
- Example: class B mask is

255.255.0.0

NOTES

Subnet Addressing

- Goal: extend address space
- Invented in 1980s
- Works within a site
- Technique
  - Assign single network prefix to site
  - Divide suffix into two parts: network at site and host
- Typical use: divide class B address

Example Of Subnet Addressing

- Single Class B number such as 128.10.0.0 assigned to site
- Site chooses subnet boundary such as 24 bits
- Routers and hosts configured with corresponding subnet mask

M = 255.255.255.0

- Given destination address, *D*, extract prefix with ‘‘logical and’’ operation

D & M

NOTES

Classless Addressing

- Goal: extend address space
- Invented in 1990s
- Works throughout Internet
- Accommodates
  - Original classful addresses
  - Subnet addresses
  - Other forms

Classless Addressing (continued)

- Technique
  - Allow arbitrary prefix size
  - Represent network address as pair

    (address , mask_size)
- Known as *Classless Inter-Domain Routing (CIDR)*

NOTES

CIDR

- Uses slash notation
- Example

  128.211.0.0/17

  means that the boundary between prefix and suffix occurs after the first 17 bits.
- Each network can be as large or small as needed (power of two)

Motivation For IP Packets

Because it can connect heterogeneous networks, a router cannot transmit a copy of a frame that arrives on one network across another. To accommodate heterogeneity, an internet must define a hardware-independent packet format.

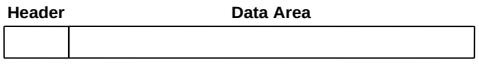
NOTES

Internet Packets

- Abstraction
- Created and understood only by software
- Contains sender and destination addresses
- Size depends on data being carried
- Called *IP datagram*

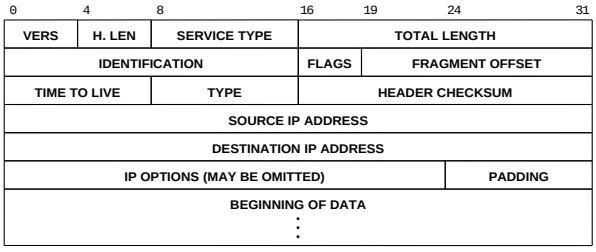
NOTES

The Two Parts Of An IP Datagram



- Header
  - Contains destination address
  - Fixed-size fields
- Payload
  - Variable size up to 64K
  - No minimum size

Datagram Header



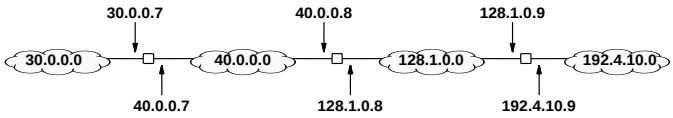
- Three key fields
  - Source IP address
  - Destination IP address
  - Type (contents)

IP Datagram Forwarding

- Performed by routers
- Similar to WAN forwarding
  - Table-driven
  - Entry specifies next hop
- Unlike WAN forwarding
  - Uses IP addresses
  - Next-hop is router or destination

NOTES

Example Of An IP Routing Table



(a)

Destination	Mask	Next Hop
30.0.0.0	255.0.0.0	40.0.0.7
40.0.0.0	255.0.0.0	deliver direct
128.1.0.0	255.255.0.0	deliver direct
192.4.10.0	255.255.255.0	128.1.0.9

(b)

- Table *(b)* is for center router in part *(a)*

Routing Table Size

Because each destination in a routing table corresponds to a network, the number of entries in a routing table is proportional to the number of networks in an internet.

NOTES

Datagram Forwarding

- Given a datagram
- Extract destination address field, *D*
- Look up *D* in routing table
- Find next-hop address, *N*
- Send datagram to *N*

Key Concept

The destination address in a datagram header always refers to the ultimate destination. When a router forwards the datagram to another router, the address of the next hop does not appear in the datagram header.

NOTES

IP Semantics

- IP is connectionless
  - Datagram contains identity of destination
  - Each datagram sent/handled independently
- Routes can change at any time

IP Semantics
(continued)

- IP allows datagrams to be
  - Delayed
  - Duplicated
  - Delivered out of order
  - Lost
- Called *best effort delivery*
- Motivation: accommodate all possible networks

NOTES

Resolving Addresses

- Hardware only recognizes MAC addresses
- IP only uses IP addresses
- Consequence: software needed to perform translation
  - Part of network interface
  - Known as *address resolution*

Address Resolution

- Layer 2 protocol
- Given
  - A locally-connected network, *N*
  - IP address *C* of computer on *N*
- Find
  - Hardware address for *C*
- Technique
  - Address Resolution Protocol

NOTES

Address Resolution Protocol (ARP)

- Keep bindings in table
- Table entry contains pair of addresses for one computer
  - IP address
  - Hardware address
- Build table automatically as needed

NOTES

ARP Table

IP Address	Hardware Address
197.15.3.2	0A:07:4B:12:82:36
197.15.3.3	0A:9C:28:71:32:8D
197.15.3.4	0A:11:C3:68:01:99
197.15.3.5	0A:74:59:32:CC:1F
197.15.3.6	0A:04:BC:00:03:28
197.15.3.7	0A:77:81:0E:52:FA

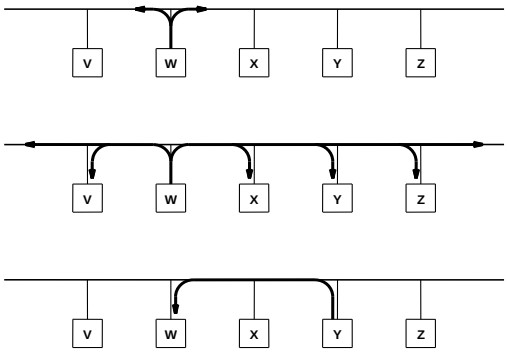
- Only contains entries for computers on local network
- IP network prefix in all entries identical

ARP Lookup Algorithm

- Look for target IP address, *T*, in ARP table
- If not found
  - Send ARP request message to *T*
  - Receive reply with *T's* hardware address
  - Add entry to table
- Return hardware address from table

NOTES

Illustration Of ARP Exchange



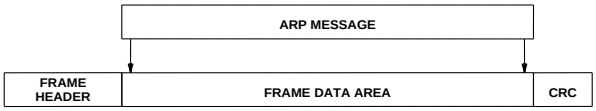
- *W* needs *Y*'s hardware address
- Request sent via broadcast
- Reply sent via unicast

ARP Message Format (For Ethernet)

0	8	16	24	31
HARDWARE ADDRESS TYPE		PROTOCOL ADDRESS TYPE		
HADDR LEN	PADDR LEN	OPERATION		
SENDER HADDR (first 4 octets)				
SENDER HADDR (last 2 octets)		SENDER PADDR (first 2 octets)		
SENDER PADDR (last 2 octets)		TARGET HADDR (first 2 octets)		
TARGET HADDR (last 4 octets)				
TARGET PADDR (all 4 octets)				

- Length of hardware address fields depend on network type
- Ethernet uses 48-bit addresses

Transmission Of ARP Message In A Frame



- ARP message sent in payload area of frame
- Called *encapsulation*

NOTES

Frame Type

Dest. Address	Source Address	Frame Type	Data In Frame
		806	complete ARP message

- Frame type identifies message as ARP
- Receiver examines frame type

Important Note

Because ARP software is part of the network interface software, all higher-layer protocols and applications can use IP addresses exclusively, and remain completely unaware of hardware addresses.

NOTES

Summary

- Internetworking
  - Solves problem of heterogeneity
  - Includes LANs and WANs
- Internet concept
  - Virtual network
  - Seamless
  - Universal

Summary (continued)

- Internet architecture
  - Multiple networks
  - Interconnected by routers
- Router
  - Special-purpose computer system
  - Interconnects two or more networks
  - Uses table to forward datagrams

NOTES

Summary (continued)

- Internet Protocol (IP)
  - Fundamental piece of TCP/IP
  - Defines
    * Internet addressing
    * Delivery semantics
    * Internet packet format (*IP datagram*)

Summary (continued)

- Address resolution
  - Needed to map IP address to equivalent hardware address
  - Part of network interface
  - Uses table
  - Automatically updates table entries
  - Broadcasts requests

NOTES

PART XI

Internetworking Part 2

(Datagram Encapsulation, Transmission, Fragmentation, Reassembly)

Internet Transmission Paradigm (General Case)

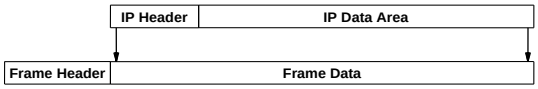
- Source host
  - Forms datagram
  - Includes destination address
  - Sends to nearest router
- Intermediate routers
  - Forward datagram to next router
- Final router
  - Delivers to destination host

NOTES

Datagram Transmission

- Datagram sent across conventional network
  - From source host and router
  - Between intermediate routers
  - From final router to destination host
- Network hardware does not recognize
  - Datagram format
  - IP addresses
- Encapsulation needed

Illustration Of IP Encapsulation



- Entire datagram treated like data
- Frame type identifies contents as IP datagram
- Frame destination address gives next hop

NOTES

Frame And Datagram Destination Addresses

- Frame address
  - Hardware (MAC) address
  - Next hop
- Datagram address
  - IP address
  - Ultimate destination

Frame Address For Encapsulated Datagram

A datagram is encapsulated in a frame for transmission across a physical network. The destination address in the frame is the address of the next hop to which the datagram should be sent; the address is obtained by translating the IP address of the next hop to an equivalent hardware address.

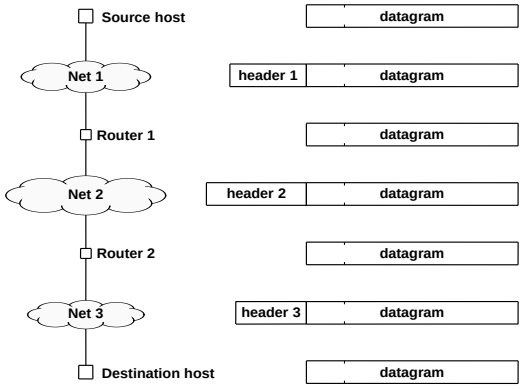
NOTES

Frames And Datagrams

- Datagram survives entire trip across Internet
- Frame only survives one hop

NOTES

Illustration Of Frame Headers Used For Datagram Transmission

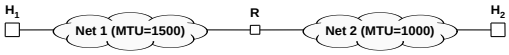


- Each hop extracts datagram and discards frame

Maximum Frame Size

- Each network technology imposes maximum frame size
  - Called *Maximum Transmission Unit (MTU)*
  - MTUs differ
- Internet
  - Can contain heterogeneous technologies
  - Must accommodate multiple MTUs

Illustration Of How Two MTUs Cause A Problem For IP



- Host *1*
  - Creates datagram for Host *2*
  - Chooses datagram size of 1500 octets
  - Transmits datagram across network *1*
- Router *R*
  - Receives datagram over network *1*
  - Must send datagram over network *2*
  - Employs *fragmentation*

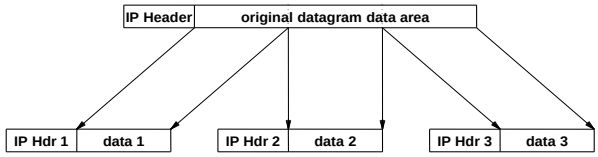
NOTES

Datagram Fragmentation

- Performed by routers
- Needed when datagram larger than MTU of network
- Divides datagram into pieces called *fragments*
- Each fragment has datagram header
- Fragments sent separately
- Ultimate destination *reassembles* fragments

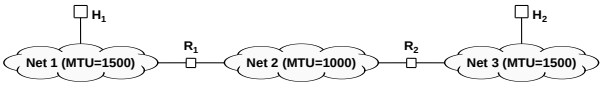
NOTES

Illustration Of Datagram Fragmentation



- Each fragment has IP datagram header
- Header fields
  - Identify original datagram
  - Indicate where fragment fits

Example Of Reassembly



- Host H_1 generates 1500-octet datagram
- Router R_1 fragments
- Router R_2 transmits fragments
- Host H_2 reassembles

Multiple Fragmenting Points

- Let MTUs along internet path be
  - 1500
  - 1500
  - 1000
  - 1500
  - 576
  - 1500
- Result: fragmentation can occur twice

NOTES

Fragmenting A Fragment

- Needed when fragment too large for network MTU
- Arbitrary subfragmentation possible
- Router divides fragments into smaller pieces
- All fragments at same ‘‘level’’
  - Offset given with respect to original datagram
  - Destination cannot distinguish subfragments

Fragment Loss

- Receiver
  - Collects incoming fragments
  - Reassembles when all fragments arrive
  - Does not know identity of router that did fragmentation
  - Cannot request missing pieces
- Consequence: Loss of one fragment means entire datagram lost

NOTES

Summary

- Internet transmission paradigm
  - Source host
  - Zero or more routers
  - Destination host
- Datagram encapsulated in network frame for transmission

Summary (continued)

- Network hardware has maximum payload size
  - Called MTU
  - Datagram must be smaller than hardware MTU
- Internet can have multiple MTUs

NOTES

Summary (continued)

- Datagram fragmentation
  - Accommodates multiple MTUs
  - Performed by router
  - Divides datagram into pieces
  - Ultimate destination reassembles

Summary
(continued)

- Fragments can be fragmented
  - Multiple levels possible
  - All offsets at one level
- Loss of any fragment means loss of entire datagram

NOTES

PART XII

Internetworking
Part 3

(Control Messages, Error Handling, ICMP)

IP Semantics

- IP is best-effort
- Datagrams can be
  - Lost
  - Delayed
  - Duplicated
  - Delivered out of order
  - Corrupted

NOTES

Error Detection

- IP does *not*
  - Introduce errors
  - Ignore all errors
- Errors detected
  - Corrupted bits
  - Illegal addresses
  - Routing loops
  - Fragment loss

Problems And Solutions

- Corrupted header bits
  - Header checksum
- Illegal destination address
  - Routing tables
- Routing loop
  - *Time-To-Live (TTL)* field
- Fragment loss
  - Timeout

NOTES

Internet Control Message Protocol (ICMP)

- Separate protocol for
  - Errors
  - Information
- Required part of IP
- Sends error messages to original source

Example ICMP Message Types

Type	Name
0	Echo Reply
1	Unassigned
2	Unassigned
3	Destination Unreachable
4	Source Quench
5	Redirect
6	Alternate Host Address
7	Unassigned
8	Echo
9	Router Advertisement
10	Router Selection
11	Time Exceeded
12	Parameter Problem
13	Timestamp Request
14	Timestamp Reply
15	Information Request
16	Information Reply
17	Address Mask Request
18	Address Mask Reply
19	Reserved (for Security)
30	Traceroute
31	Datagram Conversion Error
32	Mobile Host Redirect
33	IPv6 Where-Are-You
34	IPv6 I-Am-Here
35	Mobile Registration Request
36	Mobile Registration Reply

NOTES

Example ICMP Messages

- Source Quench
  - Sent by router
  - Triggered by datagram overrun
  - Requests sending host(s) to slow down

Example ICMP Messages (continued)

- Time Exceeded
  - Sent by router
    - TTL on datagram reached zero
    - Not a request for retransmission
  - Sent by host
    - Reassembly timeout (some fragments lost)

NOTES

Example ICMP Messages (continued)

- Destination unreachable
  - Specifies whether
    - Destination network unreachable
    - Destination host unreachable
    - Protocol port on destination unreachable

Example ICMP Messages (continued)

- Redirect
  - Sent by router
  - Goes to host on local network
  - Host used incorrect initial router
  - Requests host to change routes

NOTES

Example ICMP Messages (continued)

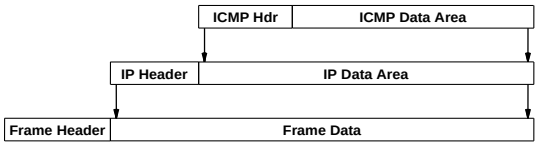
- Echo request and reply
  - Not an error
  - Tests whether destination reachable
  - Request sent by *ping* program
  - Reply sent by ICMP on destination computer

ICMP Message Transport

- Error messages go back to original source (may cross internet)
- Messages carried in IP

NOTES

Illustration Of ICMP Message Encapsulation



- Two levels of encapsulation
- IP type field specifies ICMP

Avoiding An Infinite Loop

- What happens if:
  - Datagram *D* causes an ICMP error message, I_1
  - Error message I_1 causes another error, which generates ICMP message I_2
  - Message I_2 generates another error, I_3
  - Error messages cascade
- To avoid the problem
  - No error messages about ICMP error messages

NOTES

Path MTU Discovery

- IP datagram header contains a bit to specify no fragmentation allowed
- ICMP sends an error message when fragmentation required, but not permitted
- Technique
  - Probe to find largest MTU that does not generate an error message
- Note: path MTU not guaranteed if routes change

Summary

- IP uses best-effort delivery semantics
- IP includes mechanisms to detect errors
  - Header checksum
  - Time-to-live field

NOTES

Summary
(continued)

- Internet Control Message Protocol
  - Has both error and informational messages
  - Closely integrated with IP
  - ICMP messages
    * Encapsulated in IP
    * Sent back to original source
  - Used by diagnostic programs like *ping*

NOTES

PART XIII

Internetworking Part 4

(Transport Protocols, UDP and TCP, Protocol Port Numbers)

Transport Protocol

- Separate layer of protocol stack
- Conceptually between
  - Applications
  - IP

Terminology

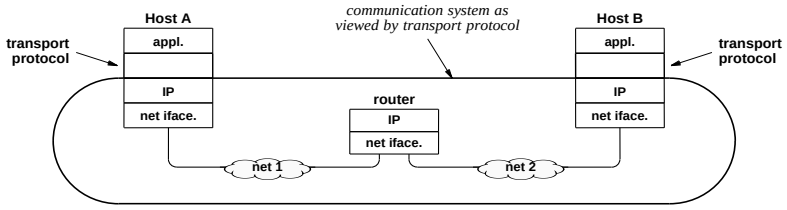
- IP
  - Provides computer-to-computer communication
  - Source and destination addresses are computers
  - Called *machine-to-machine*
- Transport protocols
  - Provide application-to-application communication
  - Need extended addressing mechanism to identify applications
  - Called *end-to-end*

NOTES

Transport Protocol Functionality

- Identify sending and receiving applications
- Optionally provide
  - Reliability
  - Flow control
  - Congestion control
- Note: not all transport protocols provide above facilities

Relationship Between Transport Protocols And Other Protocols



- Transport protocols are *end-to-end*
- Transport protocol on one computer uses IP to communicate with transport protocol on another computer

NOTES

Two Transport Protocols Available

- *Transmission Control Protocol (TCP)*
- *User Datagram Protocol (UDP)*
- Major differences
  - Interface provided to applications
  - Underlying functionality

User Datagram Protocol

- Lightweight transport
- Becoming more popular (IP telephony)
- Best-effort delivery

NOTES

UDP Features

- Connectionless service
- Arbitrary interaction
- Message-oriented interface
- Best-effort semantics
- Each message encapsulated in IP datagram
- Uses protocol ports to identify applications

UDP Details

- Accepts and delivers messages
  - Message received is exactly same as message sent
  - Boundaries preserved
- Maximum message size approximately 64K octets
- Efficient
  - No connection overhead
  - No state information maintained

NOTES

UDP Semantics

- Same best-effort semantics as IP (i.e., unreliable transfer)
- Message can be
  - Lost
  - Duplicated
  - Delayed
  - Delivered out of order
- Works best in LAN applications

Interaction With UDP

- UDP allows communication that is
  - 1-to-1
  - 1-to-many
  - Many-to-1
  - Many-to-many
- Application programmer chooses

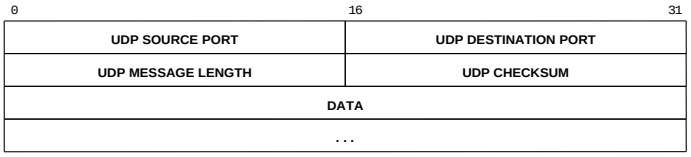
NOTES

Packet Delivery

- UDP can support
  - Unicast
  - Multicast
  - Broadcast

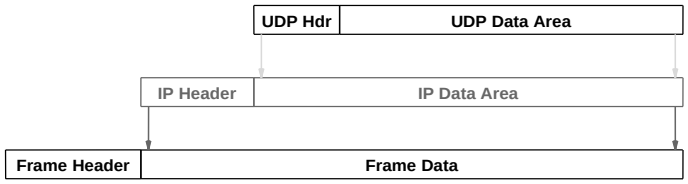
NOTES

User Datagram Format



- Extremely small header (called *thin layer*)
- Checksum optional

UDP Encapsulation



- Two levels of encapsulation
- UDP datagram size cannot exceed maximum IP payload

Identifying An Application

- Cannot extend IP address
  - No unused bits
- Cannot use OS-dependent quantity
  - Process ID
  - Task number
  - Job name
- Must work on all computer systems

NOTES

Identifying An Application
(continued)

- Invent new abstraction
  - Called *protocol port number*
  - Used to identify sending or receiving application unambiguously
  - Independent of underlying operating system
  - Used only with TCP/IP protocols

Protocol Port Numbers

- Server
  - Follows standard
  - Always uses same port number
  - Uses low port numbers
- Client
  - Obtains unused port from protocol software
  - Uses high port numbers

NOTES

Protocol Port Example

- Domain name server application is assigned port 53
- Application using DNS obtains port 28900
- UDP datagram sent from application to DNS server has
  - Source port number 28900
  - Destination port number 53
- When DNS server replies, UDP datagram has
  - Source port number 53
  - Destination port number 28900

Transmission Control Protocol (TCP)

- Major transport protocol used in Internet
- Heavily used
- Completely reliable transfer

NOTES

TCP Features

- Connection-oriented service
- Point-to-point
- Full-duplex communication
- Stream interface
- Stream divided into segments for transmission
- Each segment encapsulated in IP datagram
- Uses protocol ports to identify applications

TCP Feature Summary

TCP provides a completely reliable (no data duplication or loss), connection-oriented, full-duplex stream transport service that allows two application programs to form a connection, send data in either direction, and then terminate the connection.

NOTES

Apparent Contradiction

- IP offers best-effort (unreliable) delivery
- TCP uses IP
- TCP provides completely reliable transfer
- How is this possible?

Achieving Reliability

- Reliable connection startup
- Reliable data transmission
- Graceful connection shutdown

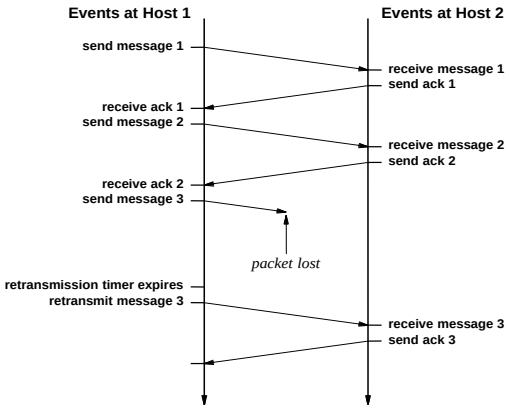
NOTES

Reliable Data Transmission

- Positive acknowledgment
  - Receiver returns short message when data arrives
  - Called *acknowledgment*
- Retransmission
  - Sender starts timer whenever message is transmitted
  - If timer expires before acknowledgment arrives, sender *retransmits* message

NOTES

Illustration Of Retransmission



How Long Should TCP Wait Before Retransmitting?

- Time for acknowledgment to arrive depends on
  - Distance to destination
  - Current traffic conditions
- Multiple connections can be open simultaneously
- Traffic conditions change rapidly

Important Point

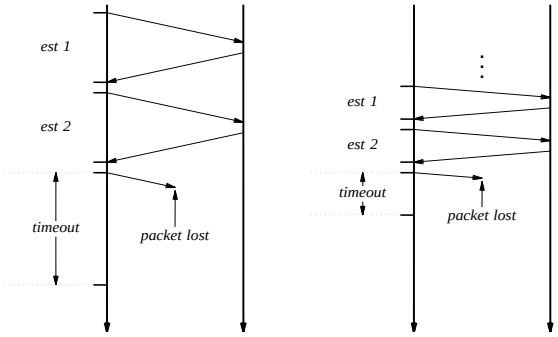
The delay required for data to reach a destination and an acknowledgment to return depends on traffic in the internet as well as the distance to the destination. Because it allows multiple application programs to communicate with multiple destinations concurrently, TCP must handle a variety of delays that can change rapidly.

NOTES

Solving The Retransmission Problem

- Keep estimate of round trip time on each connection
- Use current estimate to set retransmission timer
- Known as *adaptive retransmission*
- Key to TCP's success

Illustration Of Adaptive Retransmission



- Timeout depends on current round-trip estimate

NOTES

TCP Flow Control

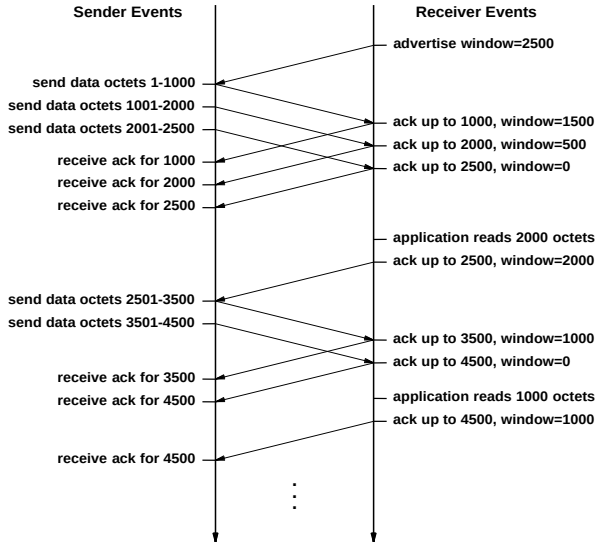
- Receiver
  - Advertises available buffer space
  - Called *window*
- Sender
  - Can send up to entire window before ack arrives

Window Advertisement

- Each acknowledgment carries new window information
  - Called *window advertisement*
  - Can be zero (called *closed window*)
- Interpretation: have received up through *X*, and can take *Y* more octets

NOTES

Illustration Of Window Advertisement



Startup And Shutdown

- Connection startup
  - Must be reliable
- Connection shutdown
  - Must be graceful
- Difficult

NOTES

Why Startup And Shutdown Are Difficult

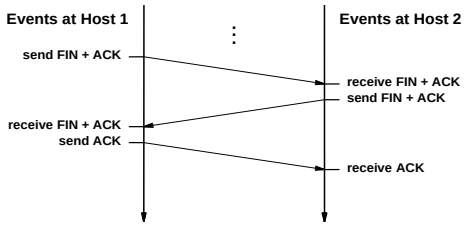
- Segments can be
  - Lost
  - Duplicated
  - Delayed
  - Delivered out of order
  - Either side can crash
  - Either side can reboot
- Need to avoid duplicate shutdown message from affecting later connection

TCP's Solution For Startup/Shutdown

- Uses three-message exchange
- Known as *3-way handshake*
- Necessary and sufficient for
  - Unambiguous, reliable startup
  - Unambiguous, graceful shutdown
- *SYN* used for startup
- *FIN* used for shutdown

NOTES

Illustration Of 3-Way Handshake



TCP Segment Format

- All TCP segments have same format
  - Data
  - Acknowledgment
  - SYN (startup)
  - FIN (shutdown)
- Segment divided into two parts
  - Header
  - Payload area (zero or more bytes of data)

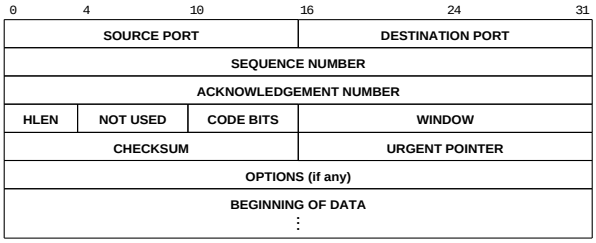
NOTES

TCP Segment Format
(continued)

- Header contains
  - Protocol port numbers to identify
    - Sending application
    - Receiving application
  - Code bits to specify items such as
    - SYN
    - FIN
    - ACK
  - Fields for window advertisement, acknowledgment, etc.

NOTES

Illustration Of TCP Segment



- Sequence number specifies where in stream outgoing data belongs
- Acknowledgment number refers to incoming data
- Few segments contain options

Network Address Translation (NAT)

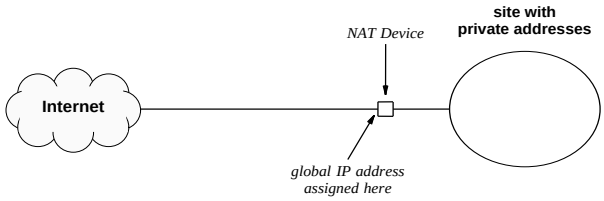
- Extension of original addressing scheme
- Motivated by exhaustion of IP address space
- Allows multiple computers to share single address
- Requires device to perform packet translation
- Implementations available
  - Stand-alone hardware device
  - IP router with NAT functionality embedded

NAT Details

- Site
  - Obtains single, valid IP address
  - Assigns a *private* address to each computer
  - Uses *NAT box* to connect to Internet
- NAT
  - Translates addresses in IP datagrams

NOTES

Illustration Of NAT



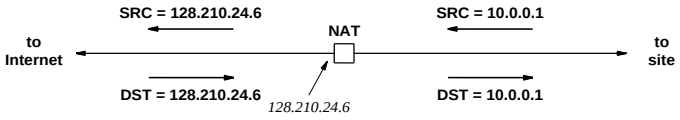
- Single valid IP address needed
- Computers at site assigned private, nonroutable addresses

NAT Example

- Site uses private network 10.0.0.0/8 internally
  - First computer assigned 10.0.0.1
  - Second computer assigned 10.0.0.2
  - And so on ...
- Site obtains valid IP address (e.g., 128.210.24.6).
- Assume computer 10.0.0.1 sends to 128.211.134.4
  - NAT translates IP source address of outgoing datagram
  - NAT translates destination address of incoming datagram

NOTES

Illustration Of NAT Translation



- Transparent to each end
  - Computer at site sends and receives datagrams normally
  - Computer in Internet receives datagrams from NAT box

NOTES

Implementation Of NAT

Direction	Field	Old Value	New Value
out	IP Source	10.0.0.1	128.10.24.6
in	IP Destination	128.10.24.6	10.0.0.1

- NAT device stores state information in table
- Value entered in table when NAT box receives outgoing datagram for new destination

Variants Of NAT

- *Basic NAT*
  - Changes IP addresses
- *Network Address and Port Translation* (*NAPT*)
  - Changes IP addresses and protocol port numbers
  - Most popular form
- *Twice NAT*
  - Used with site that runs server
  - NAT box connected to Domain Name System (DNS) server

Network Address and Port Translation (NAPT)

- By far the most popular form of NAT
- Can change TCP or UDP protocol port numbers as well as IP addresses
- Allows
  - Multiple computers at site to communicate with single destination
  - Multiple users on given computer to communicate with the same destination

NOTES

TCP Splicing

- Popular use of NAPT
- Interconnects two independent TCP connections
- Performs segment rewriting
- Extremely efficient: avoids overhead of extracting data from one connection and sending to the other
- Uses extended translation table

Example NAPT Translation Table

Direction	Fields	Old Value	New Value
out	IP SRC:TCP SRC	10.0.0.1:30000	128.10.19.20:40001
out	IP SRC:TCP SRC	10.0.0.2:30000	128.10.19.20:40002
in	IP DEST:TCP DEST	128.10.19.20:40001	10.0.0.1:30000
in	IP DEST:TCP DEST	128.10.19.20:40002	10.0.0.2:30000

- Entry in table records protocol port number as well as IP address
- Port numbers reassigned to avoid conflicts

NOTES

Summary

- Transport protocols fit between applications and Internet Protocol
- Two transport protocols in TCP/IP suite
  - User Datagram Protocol (UDP)
  - Transmission Control Protocol (TCP)
- UDP
  - Unreliable
  - Message-oriented interface

Summary (continued)

- TCP
  - Major transport protocol used in Internet
  - Complete reliability
  - Stream-oriented interface
  - Uses adaptive retransmission

NOTES

Summary (continued)

- Protocol ports
  - Integers
  - Used to identify sending and receiving applications
  - Allow unambiguous, simultaneous communication with multiple applications
- Network Address Translation
  - Allows multiple computers at site to share a single global IP address

NOTES

PART XIV

Internet Routing

(Static and automatic routing; route propagation; BGP, RIP, OSPF; multicast routing)

Terminology

- Forwarding
  - Refers to datagram transfer
  - Performed by host or router
  - Uses routing table
- Routing
  - Refers to propagation of routing information
  - Performed by routers
  - Inserts / changes values in routing table

Two Forms Of Internet Routing

- Static routing
  - Table initialized when system boots
  - No further changes
- Automatic routing
  - Table initialized when system boots
  - Routing software learns routes and updates table
  - Continuous changes possible

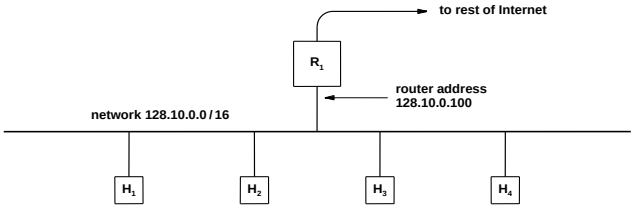
NOTES

Static Routing

- Used by most Internet hosts
- Typical routing table has two entries:
  - Local network $\rightarrow$ direct delivery
  - *Default* $\rightarrow$ nearest router

NOTES

Example Of Static Routing



(a)

Net	*Mask*	*Next hop*
128.10.0.0	255.255.0.0	direct
default	0.0.0.0	128.10.0.100

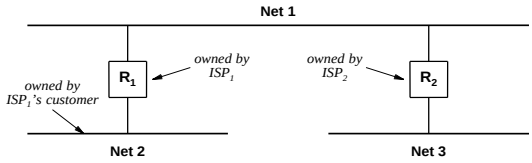
(b)

Automatic Routing

- Used by IP routers
- Requires special software
- Each router communicates with neighbors
- Passes routing information
- Uses *route propagation protocol*

NOTES

Example Of Route Propagation



- Each router *advertises* destinations that lie beyond it

The Point Of Routing Exchange

Each router runs routing software that learns about destinations other routers can reach, and informs other routers about destinations that it can reach. The routing software uses incoming information to update the local routing table continuously.

Autonomous System (AS)

- Set of networks and routers under one administrative authority
- Flexible, soft definition
- Intuition: a single corporation
- Needed because no routing protocol can scale to entire Internet
- Each AS chooses a routing protocol

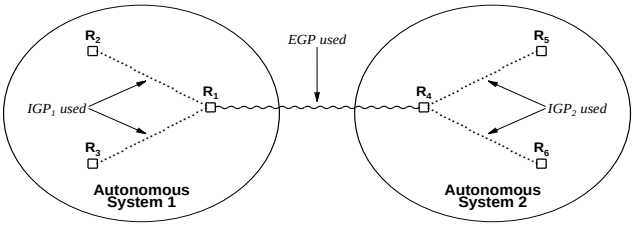
NOTES

Classifications Of Internet Routing Protocols

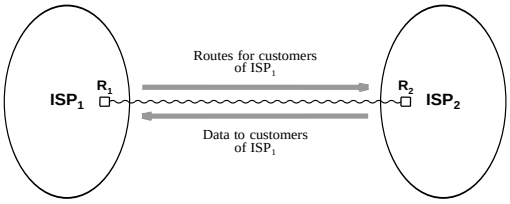
- Two broad classes
- Interior Gateway Protocols (IGPs)
  - Used among routers within autonomous system
  - Destinations lie within AS
- Exterior Gateway Protocols (EGPs)
  - Used among autonomous systems
  - Destinations lie throughout Internet

NOTES

Illustration Of IGP/EGP Use



The Concept Of Route And Data Flow



Each ISP is an autonomous system that uses an Exterior Gateway Protocol to advertise its customers' networks to other ISPs. After an ISP advertises destination, D*, datagrams destined for* D *can begin to arrive.*

Specific Internet Routing Protocols

- Border Gateway Protocol (BGP)
- Routing Information Protocol (RIP)
- Open Shortest Path First Protocol (OSPF)

NOTES

Border Gateway Protocol (BGP)

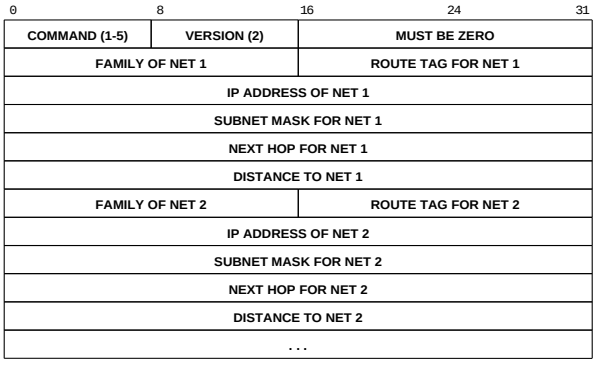
- Provides routing among autonomous systems (EGP)
- Polices to control routes advertised
- Uses reliable transport (TCP)
- Gives path of autonomous systems for each destination
- Currently, the EGP of choice in the Internet
- Current version is four (BGP-4)

The Routing Information Protocol (RIP)

- Routing within an autonomous system (IGP)
- Hop count metric
- Unreliable transport (uses UDP)
- Broadcast or multicast delivery
- Distance vector algorithm
- Can propagate a default route
- Implemented by Unix program *routed*

NOTES

Illustration Of RIP Packet Format



- Format for Version 2 (current version)

The Open Shortest Path First Protocol (OSPF)

- Routing within an autonomous system (IGP)
- Full CIDR and subnet support
- Authenticated message exchange
- Allows routes to be imported from outside the autonomous system
- Uses link-status (SPF) algorithm
- Support for multi-access networks (e.g., Ethernet)

NOTES

OSPF Areas And Efficiency

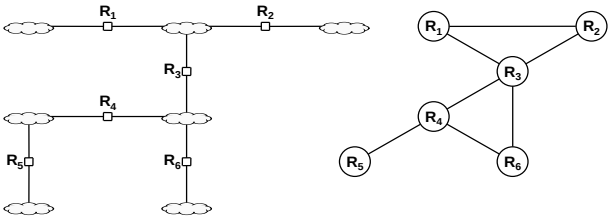
- Allows subdivision of AS into *areas*
- Link-status information propagated within area
- Routes summarized before being propagated to another area
- Reduces overhead (less broadcast traffic)

NOTES

Link-Status In The Internet

- Router corresponds to node in graph
- Network corresponds to edge
- Adjacent pair of routers periodically
  - Test connectivity
  - Broadcast link-status information to area
- Each router uses link-status messages to compute shortest paths

Illustration Of Simplified OSPF Graph



(a) *(b)*

- (a) An interconnect of routers and networks
- (b) An equivalent OSPF graph
- Router corresponds to a node in the graph
- In practice OSPF is more complex than shown

OSPF And Scale

Because it allows a manager to partition the routers and networks in an autonomous system into multiple areas, *OSPF can scale to handle a much larger number of routers than other IGPs.*

NOTES

Internet Multicast Routing

- Difficult because Internet multicast allows
  - Arbitrary computer to join multicast group at any time
  - Arbitrary member to leave multicast group at any time
  - Arbitrary computer to send message to a group (even if not a member)
- Internet Group Multicast Protocol (IGMP)
  - Used between computer and local router
  - Specifies multicast group membership

Multicast Routing Protocols

- Several protocols exist
  - Distance Vector Multicast Routing Protocol (DVMRP)
  - Core Based Trees (CBT)
  - Protocol Independent Multicast – Sparse Mode (PIM-SM)
  - Protocol Independent Multicast – Dense Mode (PIM-DM)
  - Multicast extensions to the Open Shortest Path First (MOSPF)
- None best in all circumstances

NOTES

Summary

- Static routing used by hosts
- Routers require automatic routing
- Internet divided into autonomous systems
- Two broad classes of routing protocols
  - Interior Gateway Protocols (IGPs) provide routing within an autonomous system
  - Exterior gateway Protocols (EGPs) provide routing among autonomous systems

Summary (continued)

- Border Gateway Protocol (BGP) is current EGP used in Internet
- Interior Gateway Protocols include:
  - Routing Information Protocol (RIP)
  - Open Shortest Path First protocol (OSPF)
- Internet multicast routing difficult
  - Protocols proposed include: DVMRP, PIM-SM, PIM-DM, MOSPF

NOTES

PART XV

Internet Applications

(Client-Server Concept, Use of Protocol Ports, Socket API, DNS, E-mail, VoIP, TELNET, FTP)

Functionality

- Transport layer and layers below
  - Basic communication
  - Reliability
- Application layer
  - Abstractions
    - Files
    - Services
    - Databases
  - Names

NOTES

Dichotomy Of Duties

- Network
  - Transfers bits
  - Operates at application's request
- Applications determine
  - What to send
  - When to send
  - Where to send
  - Meaning of bits

Important Point

Although an internet system provides a basic communication service, the protocol software cannot initiate contact with, or accept contact from, a remote computer. Instead, two application programs must participate in any communication: one application initiates communication and the other accepts it.

NOTES

How Two Application Programs Make Contact

- One application
  - Begins execution first
  - Waits passively at prearranged location
- Another application
  - Begins execution later
  - Actively contacts first program
- Called *client-server interaction*

Client-Server Paradigm

- Used by all network applications
- Passive program called a *server*
- Active program called a *client*

NOTES

Internet Communication

All network applications use a form of communication known as the client-server paradigm. *A server application waits passively for contact, while a client application initiates communication actively.*

Characteristics Of A Client

- Arbitrary application program
- Becomes client temporarily
- Can also perform other computations
- Invoked directly by user
- Runs locally on user's computer
- Actively initiates contact with a server
- Contacts one server at a time

NOTES

Characteristics Of A Server

- Special-purpose, privileged program
- Dedicated to providing one service
- Can handle multiple remote clients simultaneously
- Invoked automatically when system boots
- Executes forever
- Needs powerful computer and operating system
- Waits passively for client contact
- Accepts requests from arbitrary clients

Terminology

- *Server*
  - An executing program that accepts contact over the network
- *Server-class computer*
  - Hardware sufficient to execute a server
- Informally
  - Term *server* often applied to computer

NOTES

Direction Of Data Flow

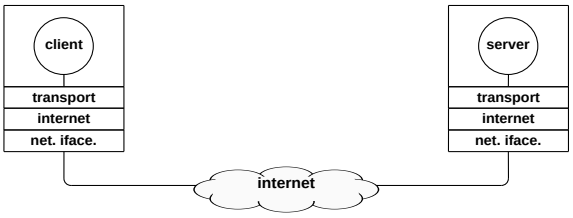
- Data can flow
  - From client to server only
  - From server to client only
  - In both directions
- Application protocol determines flow
- Typical scenario
  - Client sends request(s)
  - Server sends response(s)

Key Idea

Although the client initiates contact, information can flow in either or both directions between a client and server. Many services arrange for the client to send one or more requests and the server to return one response for each request.

NOTES

Clients, Servers, And Other Protocols



- Clients and servers are application programs

Server CPU Use

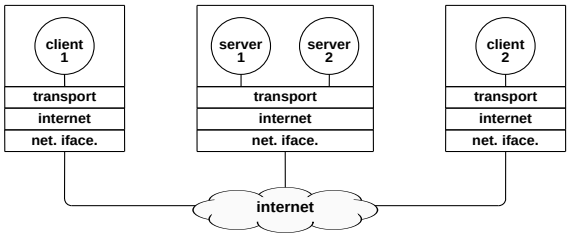
- Facts
  - Server operates like other applications
    * Uses CPU to execute instructions
    * Performs I/O operations
  - Waiting for data to arrive over a network does not require CPU time
- Consequence
  - Server program only uses CPU when servicing a request

NOTES

Multiple Services

- Can have multiple servers on single computer
- Servers only use processor when handling a request
- Powerful hardware required to handle many services simultaneously

Illustration Of Multiple Servers



- Each server offers one service
- One server can handle multiple clients

NOTES

Identifying A Service

- Protocol port number used
- Each service given unique port number, *P*
- Server
  - Informs OS it is using port *P*
  - Waits for requests to arrive
- Client
  - Forms request
  - Sends request to port *P* on server computer

The Point About Ports

Transport protocols assign each service a unique port identifier. A server must specify the identifier when it begins execution. A client must specify the identifier when it requests transport protocol software to contact a server. Protocol software on the server computer uses the identifier to direct an incoming request to the correct server.

NOTES

In Theory

- Port numbers are merely integers
- Any server could use any port number

In Practice

- Protocol port numbers are used as service identifiers
- Need uniform numbering
  - To allow arbitrary client to contact server on arbitrary machine
  - To avoid inventing ‘‘directory assistance’’ mechanism
- Port numbers
  - Uniform throughout Internet
  - Set by standards bodies

NOTES

Terminology

- *Sequential program*
  - Typical of most programs
  - Single thread of control
- *Concurrent program*
  - Multiple threads of control
  - Execution proceeds ‘‘in parallel’’
  - More difficult to create

Servers And Concurrency

- Sequential server
  - Also called *iterative*
  - Handles one request at a time
- Concurrent server
  - Can handle multiple requests at a time
  - No waiting

NOTES

Delay In Servers

- Concurrent server
  - Server creates new thread of control to handle each request
  - Client only waits for its request to be processed
- Sequential server
  - Client waits for all previous requests to be processed as well as for its request to be processed
  - Unacceptable to user if long request blocks short request

Concurrency In Servers

Concurrent execution is fundamental to servers because concurrency permits multiple clients to obtain a given service without having to wait for the server to finish previous requests. In a concurrent server, the main server thread creates a new service thread to handle each client.

NOTES

Protocol Ports And Concurrent Servers

- Apparent problem
  - One port number assigned to each service
  - Concurrent server has multiple copies (threads) running
  - Client and server can interact
  - Messages sent to server's port must be delivered to correct copy of server

Protocol Ports And Concurrent Servers (continued)

- Solution to problem: use information about client as well as server to deliver incoming packets
- TCP uses four items to identify connection
  - Server's IP address
  - Server's protocol port number
  - Client's IP address
  - Client's protocol port number

NOTES

Demultiplexing In A Concurrent Server

Transport protocols assign an identifier to each client as well as to each service. Protocol software on the server's machine uses the combination of client and server identifiers to choose the correct copy of a concurrent server.

Variations On A Theme

- A server can use
  - Connectionless transport (UDP)
  - Connection-oriented transport (TCP)
  - Both for a single service
- A single server can offer multiple services
  - Often used for trivial services
  - Server uses multiple port numbers simultaneously

NOTES

Variations On A Theme
(continued)

- A server can
  - Maintain interaction with a client for days or hours
  - Send a short response and terminate interaction
  - Perform I/O on the local computer
  - Become a client for another service (potential cycle problem)

Example Of Circularity

- Time server
  - Returns time of day
- File server
  - Allows client to read or write a file
  - Calls time server when generating time stamp for file
- Suppose programmer modifies time server to log requests to a file

NOTES

Interacting With Protocol Software

- Client or server uses transport protocols
- Protocol software inside OS
- Applications outside OS
- Mechanism needed to bridge the two
  - Called *Application Program Interface (API)*

Application Program Interface

- Part of operating system
- Permits application to use protocols
- Defines
  - Operations allowed
  - Arguments for each operation

NOTES

Socket API

- Originally designed
  - For BSD UNIX
  - To use with TCP/IP protocols
- Now
  - Industry standard
  - Available on many operating systems

Socket

- OS Abstraction (not hardware)
- Created dynamically
- Persists only while application runs
- Referenced by a descriptor

NOTES

Descriptor

- Small integer
- One per active socket
- Used in all operations on socket
- Generated by OS when socket created
- Only meaningful to application that owns socket
- In UNIX, integrated with file descriptors

Creating A Socket

- Application calls *socket* function

```
sdesc = socket(protofamily, type, proto)
```

- OS returns descriptor for socket
- Descriptor valid until application closes socket or exits

NOTES

Socket Functionality

- Socket completely general
- Can be used
  - By client
  - By server
  - With a CO transport protocol
  - With a CL transport protocol
  - To send data, receive data, or both
- Large set of operations

Socket Operations

- *Close*
  - Terminate use of socket
  - Permanent
- *Bind*
  - Specify protocol port for a socket
  - Specify local IP address for a socket
  - Can use *INADDR_ANY* for any IP address

NOTES

Socket Operations
(continued)

- *Listen*
  - Used by server
  - Prepares socket to accept incoming connections
- *Accept*
  - Used by server
  - Waits for next connection and returns new socket

Socket Operations
(continued)

- *Connect*
  - Used by client
  - Either
    * Forms a TCP connection
    * Fully specifies addresses for UDP

NOTES

Two Purposes Of The Connect Function

The connect *function, which is called by clients, has two uses. With connection-oriented transport,* connect *establishes a transport connection to a specified server. With connectionless transport,* connect *records the server's address in the socket, allowing the client to send many messages to the same server without specifying the destination address with each message.*

Socket Operations
(continued)

- *Send*, *sendto*, and *sndmsg*
  - Transfer outgoing data from application
- *Recv*, *recvfrom*, and *recvmsg*
  - Transfer incoming data to application
- Many additional functions
  - Supply support and utility services
  - Some implemented as library calls

NOTES

Examples Of Socket Support Functions

- *Gethostbyname*
  - Maps domain name to IP address
  - Example of argument

    `"www.netbook.cs.purdue.edu"`
- *Getprotobyname*
  - Maps name of protocol to internal number
  - Argument usually `"tcp"` or `"udp"`

An Example Service

- Purpose
  - Count times invoked
  - Return printable ASCII message
- Connection-oriented protocol
- Sequential execution (not concurrent)

NOTES

An Example Client

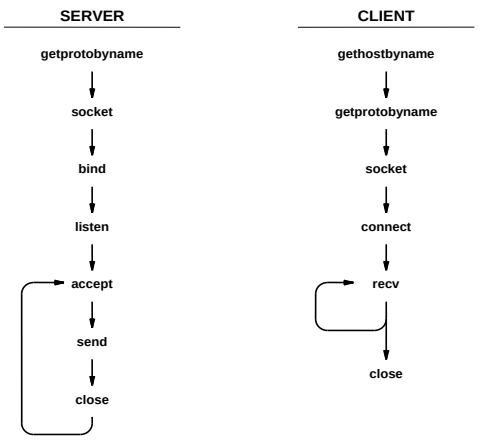
- Open TCP connection to server
- Iterate until end-of-file
  - Receive text
  - Print characters received
- Close connection
- Exit

NOTES

Example Server

- Create socket and put in passive mode
- Iterate forever
  - Accept next connection, get new socket
  - Increment count and send text message
  - Close socket for connection
- Notes
  - Main socket remains open
  - Server never exits

Socket Calls In Client And Server



- Client closes socket after use
- Server never closes original socket

Code For Client

- Arguments to program
  - Host
  - Protocol port
  - Both optional
- Many details
- Minor incompatibilities among socket implementations
  - Unix
  - Microsoft
  - Use C #ifdef

NOTES

Example Client Code (1)

```
/* client.c - code for example client program that uses TCP */

#ifndef unix
#define WIN32
#include <windows.h>
#include <winsock.h>
#else
#define closesocket close
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#endif

#include <stdio.h>
#include <string.h>

#define PROTOPORT       5193            /* default protocol port number */

extern  int             errno;
char    localhost[] =   "localhost";    /* default host name            */
/*------------------------------------------------------------------------
 * Program:   client
 *
 * Purpose:   allocate a socket, connect to a server, and print all output
```

Example Client Code (2)

```
 * Syntax:    client [ host [port] ]
 *
 *              host  - name of a computer on which server is executing
 *              port  - protocol port number server is using
 *
 * Note:      Both arguments are optional.  If no host name is specified,
 *            the client uses "localhost"; if no protocol port is
 *            specified, the client uses the default given by PROTOPORT.
 *
 *------------------------------------------------------------------------
 */
main(argc, argv)
int     argc;
char    *argv[];
{
        struct  hostent  *ptrh;  /* pointer to a host table entry       */
        struct  protoent *ptrp;  /* pointer to a protocol table entry   */
        struct  sockaddr_in sad; /* structure to hold an IP address     */
        int     sd;              /* socket descriptor                   */
        int     port;            /* protocol port number                */
        char    *host;           /* pointer to host name                */
        int     n;               /* number of characters read           */
        char    buf[1000];       /* buffer for data from the server     */
#ifdef WIN32
        WSADATA wsaData;
        WSAStartup(0x0101, &wsaData);
#endif
        memset((char *)&sad,0,sizeof(sad)); /* clear sockaddr structure */
        sad.sin_family = AF_INET;         /* set family to Internet     */
```


NOTES

Example Client Code (3)

```
        /* Check command-line argument for protocol port and extract    */
        /* port number if one is specified.  Otherwise, use the default */
        /* port value given by constant PROTOPORT                       */

        if (argc > 2) {                 /* if protocol port specified   */
                port = atoi(argv[2]);   /* convert to binary            */
        } else {
                port = PROTOPORT;       /* use default port number      */
        }
        if (port > 0)                   /* test for legal value         */
                sad.sin_port = htons((u_short)port);
        else {                          /* print error message and exit */
                fprintf(stderr,"bad port number %s\n",argv[2]);
                exit(1);
        }

        /* Check host argument and assign host name. */

        if (argc > 1) {
                host = argv[1];         /* if host argument specified   */
        } else {
                host = localhost;
        }
```

Example Client Code (4)

```
        /* Convert host name to equivalent IP address and copy to sad. */

        ptrh = gethostbyname(host);
        if ( ((char *)ptrh) == NULL ) {
                fprintf(stderr,"invalid host: %s\n", host);
                exit(1);
        }
        memcpy(&sad.sin_addr, ptrh->h_addr, ptrh->h_length);

        /* Map TCP transport protocol name to protocol number. */

        if ( ((int)(ptrp = getprotobyname("tcp"))) == 0) {
                fprintf(stderr, "cannot map \"tcp\" to protocol number");
                exit(1);
        }

        /* Create a socket. */

        sd = socket(PF_INET, SOCK_STREAM, ptrp->p_proto);
        if (sd < 0) {
                fprintf(stderr, "socket creation failed\n");
                exit(1);
        }

        /* Connect the socket to the specified server. */

        if (connect(sd, (struct sockaddr *)&sad, sizeof(sad)) < 0) {
                fprintf(stderr,"connect failed\n");
                exit(1);
        }
```


NOTES

Example Client Code (5)

```
        /* Repeatedly read data from socket and write to user's screen. */

        n = recv(sd, buf, sizeof(buf), 0);
        while (n > 0) {
                write(1,buf,n);
                n = recv(sd, buf, sizeof(buf), 0);
        }

        /* Close the socket. */

        closesocket(sd);

        /* Terminate the client program gracefully. */

        exit(0);
}
```

Code For Server

- Arguments to program
  - Protocol port
- C language #ifdefs for socket variants

NOTES

Example Server Code (1)

```
/* server.c - code for example server program that uses TCP */
#ifndef unix
#define WIN32
#include <windows.h>
#include <winsock.h>
#else
#define closesocket close
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#endif

#include <stdio.h>
#include <string.h>

#define PROTOPORT       5193            /* default protocol port number */
#define QLEN            6               /* size of request queue        */

int     visits      =   0;              /* counts client connections    */
/*------------------------------------------------------------------------
 * Program:   server
 *
 * Purpose:   allocate a socket and then repeatedly execute the following:
 *              (1) wait for the next connection from a client
 *              (2) send a short message to the client
 *              (3) close the connection
 *              (4) go back to step (1)
```

Example Server Code (2)

```
 * Syntax:    server [ port ]
 *
 *              port  - protocol port number to use
 *
 * Note:      The port argument is optional.  If no port is specified,
 *            the server uses the default given by PROTOPORT.
 *
 *------------------------------------------------------------------------
 */
main(argc, argv)
int     argc;
char    *argv[];
{
        struct  hostent  *ptrh;  /* pointer to a host table entry       */
        struct  protoent *ptrp;  /* pointer to a protocol table entry   */
        struct  sockaddr_in sad; /* structure to hold server's address  */
        struct  sockaddr_in cad; /* structure to hold client's address  */
        int     sd, sd2;         /* socket descriptors                  */
        int     port;            /* protocol port number                */
        int     alen;            /* length of address                   */
        char    buf[1000];       /* buffer for string the server sends  */

#ifdef WIN32
        WSADATA wsaData;
        WSAStartup(0x0101, &wsaData);
#endif
        memset((char *)&sad,0,sizeof(sad)); /* clear sockaddr structure */
        sad.sin_family = AF_INET;         /* set family to Internet     */
        sad.sin_addr.s_addr = INADDR_ANY; /* set the local IP address   */
```


NOTES

Example Server Code (3)

```
        /* Check command-line argument for protocol port and extract    */
        /* port number if one is specified.  Otherwise, use the default */
        /* port value given by constant PROTOPORT                       */

        if (argc > 1) {                 /* if argument specified        */
                port = atoi(argv[1]);   /* convert argument to binary   */
        } else {
                port = PROTOPORT;       /* use default port number      */
        }
        if (port > 0)                   /* test for illegal value       */
                sad.sin_port = htons((u_short)port);
        else {                          /* print error message and exit */
                fprintf(stderr,"bad port number %s\n",argv[1]);
                exit(1);
        }

        /* Map TCP transport protocol name to protocol number */

        if ( ((int)(ptrp = getprotobyname("tcp"))) == 0) {
                fprintf(stderr, "cannot map \"tcp\" to protocol number");
                exit(1);
        }

        /* Create a socket */

        sd = socket(PF_INET, SOCK_STREAM, ptrp->p_proto);
        if (sd < 0) {
                fprintf(stderr, "socket creation failed\n");
                exit(1);
        }
```

Example Server Code (4)

```
        /* Bind a local address to the socket */

        if (bind(sd, (struct sockaddr *)&sad, sizeof(sad)) < 0) {
                fprintf(stderr,"bind failed\n");
                exit(1);
        }

        /* Specify size of request queue */

        if (listen(sd, QLEN) < 0) {
                fprintf(stderr,"listen failed\n");
                exit(1);
        }

        /* Main server loop - accept and handle requests */

        while (1) {
                alen = sizeof(cad);
                if ( (sd2=accept(sd, (struct sockaddr *)&cad, &alen)) < 0) {
                        fprintf(stderr, "accept failed\n");
                        exit(1);
                }
                visits++;
                sprintf(buf,"This server has been contacted %d time%s\n",
                        visits,visits==1?".":"s.");
                send(sd2,buf,strlen(buf),0);
                closesocket(sd2);
        }
}
```


NOTES

Stream Interface

- Sender
  - Calls *send* repeatedly
  - Specifies number of octets per call
- TCP
  - Divides stream into segments
- Receiver
  - Calls *recv* repeatedly
  - Receives one or more octets per call
  - Count of zero means ‘‘end of file’’
  - Size received unrelated to size sent

Internet Applications

- Domain Name System
- Electronic mail
- IP telephony
- Remote login
- File transfer
- All use client-server model

NOTES

Names

- Internet communication requires IP addresses
- Humans prefer to use computer names
- Automated system available to translate names to addresses
- Known as *Domain Name System (DNS)*

DNS Functionality

- Given
  - Name of a computer
- Returns
  - Computer's Internet address
- Method
  - Distributed lookup
  - Client contacts server(s) as necessary

NOTES

Domain Name Syntax

- Alphanumeric *segments* separated by dots
- Examples

```
www.netbook.cs.purdue.edu

www.eg.bucknell.edu
```

- Most significant part on right

NOTES

Obtaining A Domain Name

- Organization
  - Chooses desired name
  - Must be unique
  - Registers with central authority
  - Placed under one *top-level domain*
- Names subject to international law for
  - Trademarks
  - Copyright

Original Top-Level Domains

Domain Name	*Assigned To*
com	*Commercial organization*
edu	*Educational institution*
gov	*Government organization*
mil	*Military group*
net	*Major network support center*
org	*Organization other than those above*
arpa	*Temporary ARPA domain (still used)*
int	*International organization*
country code	*A country*

- Meaning assigned to each
- Three domains considered generic

```
.com
.net
.org
```

New Top-Level Domains

Domain Name	Assigned To
aero	Air-Transport Industry
biz	Businesses
coop	Non-Profit Cooperatives
info	Unrestricted
museum	Museums
name	Individuals
pro	Professionals (accountants, lawyers, physicians)

- Added because proponents claimed *.com* insufficient

NOTES

Within Organization

- Subdivision possible
- Arbitrary levels allowed
- Not standardized
- Controlled locally by organization

Example Name Structure

- First level is *.com*
- Second level is company name
- Third level is division within company
- Fourth level either
  - Company subdivision
  - Individual computer

NOTES

An Example

- Assume
  - Company is *Foobar*
  - Has two divisions
    * Soap division
    * Candy division
- Candy division has subdivisions
- Soap division has no subdivisions

NOTES

An Example
(continued)

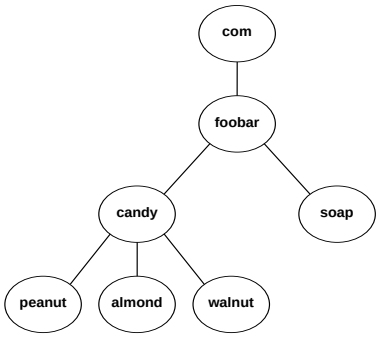
- Names in soap division have form

  computer . soap . foobar . com

- Names in candy division have form

  computer . *subdivision* . candy . foobar . com

Illustration Of Foobar Naming Hierarchy



The Point About Names

The number of segments in a domain name corresponds to the naming hierarchy. There is no universal standard; each organization can choose how to structure names in its hierarchy. Furthermore, names within an organization do not need to follow a uniform pattern; individual groups within the organization can choose a hierarchical structure that is appropriate for the group.

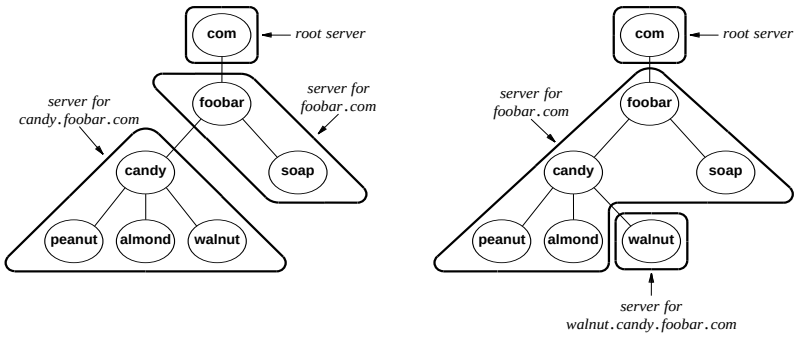
NOTES

DNS Client-Server Interaction

- Client known as *resolver*
- Multiple DNS servers used
- Arranged in hierarchy
- Each server corresponds to contiguous part of naming hierarchy

NOTES

Two Possible DNS Hierarchies



- Choice made by organization

Inter-Server Links

All domain name servers are linked together to form a unified system. Each server knows how to reach a root server and how to reach servers that are authorities for names further down the hierarchy.

In Practice

- DNS uses backup server(s)
- ISPs and others
  - Offer DNS service to subscribers
- Small organizations and individuals
  - Only need domain names for computers running servers
  - Contract with an ISP for domain service

NOTES

DNS Lookup

- Application
  - Becomes DNS client
  - Sends request to local DNS server
- Local server
  - If answer known, returns response
  - If answer unknown
    * Starts at top-level server
    * Follows links
    * Returns response
- Called *name resolution*

Caching In DNS

- Server always caches answers
- Host can cache answers
- Caching
  - Improves efficiency
  - Eliminates unnecessary search
  - Works well because high locality of reference

NOTES

DNS Types

- Each entry in server consists of
  - Domain name
  - DNS type for name
  - Value to which name corresponds
- During lookup, client must supply
  - Name
  - Type
- Server
  - Matches both name and type

The Point About Types

The Domain Name System stores a type with each entry. When a resolver looks up a name, the resolver must specify the type that is desired; a DNS server returns only entries that match the specified type.

NOTES

Example DNS Types

- Type *A (Address)*
  - Value is IP address for named computer
- Type *MX (Mail eXchanger)*
  - Value is IP address of computer with mail server for name
- Type *CNAME (Computer NAME)*
  - Value is another domain name
  - Used to establish alias (*www*)

Domain Name Abbreviation

- DNS lookup uses full names
- Users desire abbreviations
- Technique
  - Configure resolver with list of suffixes
  - Try suffixes one at a time

NOTES

Example Of DNS Abbreviation

- Suffixes are
  - *cs . purdue . edu*
  - *purdue . edu*
  - *ecn . purdue . edu*
- User enters name *www*
- Resolver tries
  - *www*
  - *www . cs . purdue . edu*
  - *www . purdue . edu*
  - *www . ecn . purdue . edu*

Other Internet Applications

- Invoked directly by user
  - E-mail
  - IP telephony
  - Remote login
  - File transfer
  - Web browsing

NOTES

Electronic Mail

- Originally
  - Memo sent from one user to another
- Now
  - Memo sent to one or more *mailboxes*
- Mailbox
  - Destination point for messages
  - Can be storage or program
  - Given unique address

E-mail Address

- Text string
- Specifies mail destination
- General form

mailbox@computer

- *computer*
  - Domain name of computer
  - Actually type MX
- *mailbox*
  - Destination on the computer

NOTES

Use Of E-mail Address

Each electronic mailbox has a unique address, which is divided into two parts: the first identifies a user's mailbox, and the second identifies a computer on which the mailbox resides. E-mail software on the sender's computer uses the second part to select a destination; e-mail software on the recipient's computer uses the first part to select a particular mailbox.

Mail Message Format

- Header
  - Identifies sender, recipient(s), memo contents
  - Lines of form

    keyword : information

- Blank line
- Body
  - Contains text of message

NOTES

Example E-mail Header Fields

Keyword	Meaning
From	Sender's address
To	Recipients' addresses
Cc	Addresses for carbon copies
Date	Date on which message was sent
Subject	Topic of the message
Reply-To	Address to which reply should go
X-Charset	Character set used (usually ASCII)
X-Mailer	Mail software used to send the message
X-Sender	Duplicate of sender's address
X-Face	Encoded image of the sender's face

- Most header lines optional

Extending E-mail

- Original e-mail
  - Message restricted to ASCII text
- Users desire to send
  - Image files
  - Audio clips
  - Compiled (binary) programs
- Solution
  - *Multi-purpose Internet Mail Extensions (MIME)*

NOTES

MIME

- Allows transmission of
  - Binary data
  - Multimedia files (video/audio clips)
  - Multiple types in single message
  - Mixed formats
- Backward compatible

MIME Encoding

- Sender
  - Inserts additional header lines
  - Encodes binary data in (printable) ASCII
- Sent like standard message
- Receiver
  - Interprets header lines
  - Extracts and decodes parts
- Separate standards for content and encoding

NOTES

Example Of MIME

- Header lines added

```
MIME-Version: 1.0
Content-Type: Multipart/Mixed; Boundary=xxxsep
```

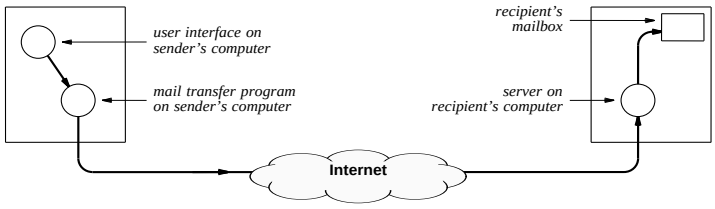
- Specifies
  - Using MIME Version *1.0*
  - Line *xxxsep* appears before each message part

NOTES

Mail Transfer

- Protocol is *Simple Mail Transfer Protocol (SMTP)*
- Runs over TCP
- Used between
  - Mail transfer program on sender's computer
  - Mail server on recipient's computer
- Specifies how
  - Client interacts with server
  - Recipients specified
  - Message is transferred

Illustration Of Mail Transfer



- Server
  - Required to receive mail
  - Places message in user's mailbox

Terminology

- *Mail exploder*
  - Program
  - Accepts incoming message
  - Delivers to multiple recipients
- *Mailing list*
  - Database
  - Used by exploder
- *Mail gateway*
  - Connects two mail systems

NOTES

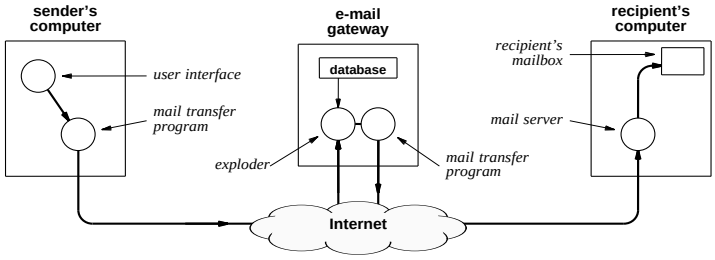
Illustration Of A Mailing List

List	Contents
friends	**Joe@foo.com, Jill@bar.gov, Tim@StateU.edu**
	Mary@acollege.edu, Hank@nonexist.com,
customers	**george@xyz.com, VP_Marketing@news.com**
bball-interest	**hank@none.com, Linda_S_Smith@there.com,**
	John_Q_Public@foobar.com, Connie@foo.edu

- Separate permissions for
  - Mailing to list
  - Adding / deleting members
    - *Public* – anyone can join
    - *Private* – access restricted by owner

NOTES

Illustration Of A Mail Gateway



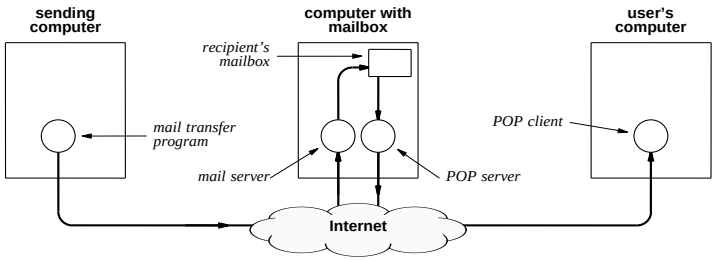
- Can connect two
  - Heterogeneous systems
  - Internet to non-Internet

Computers Without Mail Servers

- Typically
  - Small, personal computer
  - Not continuously connected to Internet
- To receive e-mail, user must
  - Establish mailbox on large computer
  - Access mailbox as necessary
- *Post Office Protocol (POP)* used

NOTES

Illustration Of POP



- Current version named *POP3*

IP Telephony

- Send audio telephone communication over an IP network
- Network types
  - Local IP network (e.g., within an organization)
  - Global Internet
- Scope of communication
  - Work entirely within an IP network
  - Interoperate with *Public Switched Telephone Network (PSTN)*
- Also known as *Voice over IP (VoIP)*

IP Telephony Standards

- Two competing groups
  - Internet Engineering Task Force (IETF) sets TCP/IP standards
  - International Telecommunications Union (ITU) controls telephone standards
- Several proposed standards
  - Session Initiation Protocol (SIP) from IETF
  - H.323 from ITU
  - Megaco from IETF and ITU

NOTES

Encoding And Transmission

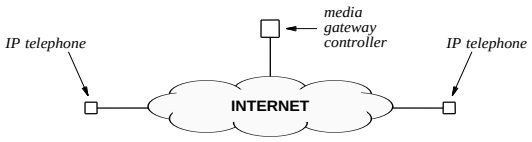
- General agreement
- Digital encoding
  - Use Pulse Code Modulation (PCM) standard
  - Same as existing telephone system
- Transmission
  - Use Real-time Transport Protocol (RTP)
  - Encapsulate RTP message in UDP datagram
  - Encapsulate UDP datagram in IP datagram

Signaling

- Telco term
- Refers to call setup, monitoring, and teardown
- Current telephone system standard is *Signaling System 7 (SS7)*
- May involve
  - Location of called party
  - Deciding whether to accept incoming call
  - Authentication and services such as caller ID
  - Accounting
- Requires active mechanism

NOTES

Illustration Of Basic IP Telephone System



- *Media gateway controller* handles signaling
- Once call is established, IP telephones can communicate directly

Interconnection With PSTN

- Difficult
- Must handle many possibilities
  - Calls that originate on IP telephone system and terminate on PSTN
  - Calls that originate on PSTN and terminate on IP telephone system
  - Mobile telephones
  - Services such as call forwarding
  - Islands (e.g., PSTN between two IP telephone systems)

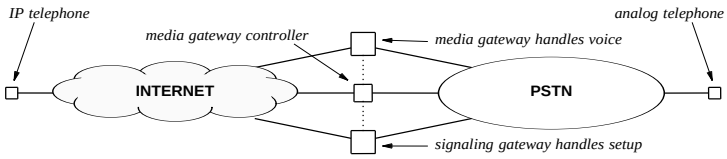
NOTES

Interconnection With PSTN (continued)

- Three conceptual functions
  - Media gateway passes encoded voice
  - Signaling gateway handles call setup
  - Media gateway controller coordinates

NOTES

Illustration Of PSTN Interconnection



- Questions
  - What individual functions should be provided?
  - How should functions be divided among gateways?
- Each proposed standard defines a set of conceptual functions and implementation mechanisms

SIP Characteristics

- Distributed signaling system
- Uses DNS to store location information
- Defines multiple servers
  - User agent
  - Location server
  - Proxy server
  - Redirect server
  - Registrar server

H.323 Characteristics

- Uses more centralized model (similar to current telephone system)
- Defines
  - Terminal
  - Gatekeeper
  - Gateway
  - Multipoint Control Unit (to handle conferencing)

NOTES

Examples Of Layering: H.323

<table>
<tr><th>Layer</th><th>Signaling</th><th>Registration</th><th>Audio</th><th>Video</th><th>Data</th><th>Security</th></tr>
<tr><td>5</td><td>H.225.0-Q.931
H.250-Annex G
H.245
H.250</td><td>H.225.9-RAS</td><td>G.711
H.263
G.722
G.723
G.728</td><td>H.261
H.323</td><td>T.120</td><td>H.235</td></tr>
<tr><td></td><td></td><td></td><td colspan="2">RTP, RTCP</td><td></td><td></td></tr>
<tr><td>4</td><td>TCP, UDP</td><td colspan="3">UDP</td><td>TCP</td><td>TCP, UDP</td></tr>
<tr><td>3</td><td colspan="6">IP, RSVP, and IGMP</td></tr>
</table>

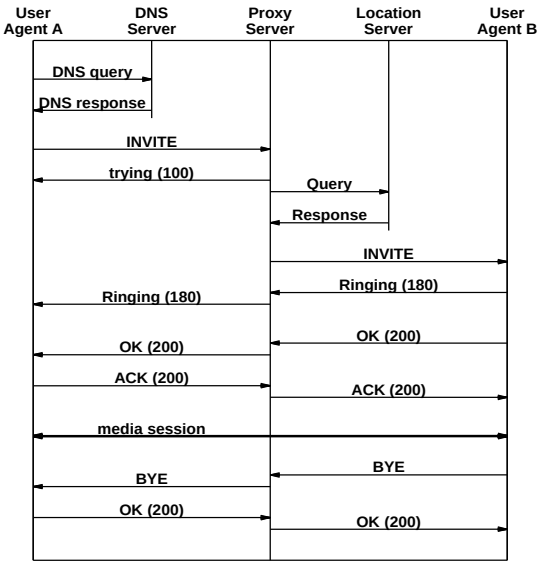
- Many protocols
- Allows data and video to be transferred along with audio

NOTES

Example Methods: SIP

Method	Purpose
INVITE	Session creation: an endpoint is invited to participate in the session.
ACK	Acknowledgment response to INVITE.
BYE	Session termination: call is ended.
CANCEL	Pending request cancellation (no effect if request has been completed).
REGISTER	Registration of user's location (i.e., a URL at which the user can be reached).
OPTIONS	Query to determine capabilities of called party.

Example SIP Session



Remote Login

- Provide interactive access to computer from remote site
- Standard protocol is *TELNET*

NOTES

TELNET

- Text-oriented interface
- User
  - Invokes client
  - Specifies remote computer
- Client
  - Forms TCP connection to server
  - Passes keystrokes over connection
  - Displays output on screen

File Transfer

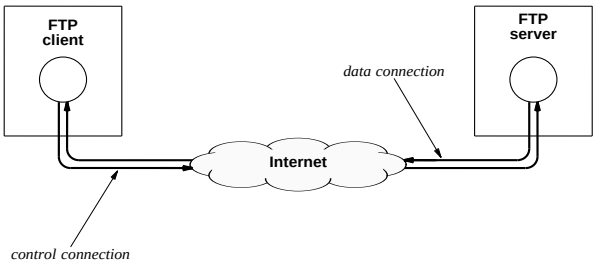
- Complete file copy
- Major protocol is *File Transfer Protocol (FTP)*
  - Uses TCP
  - Supports binary or text transfers
  - Large set of commands
  - Until 1995 was major source of packets in Internet

NOTES

FTP Paradigm

- Command-line interface
- User
  - Forms TCP connection to server (called *control connection*)
  - Logs in
  - Enters commands to list directories, transfer files
- Server
  - Establishes new TCP connection for each transfer

Illustration Of TCP Connections During An FTP File Transfer



- Two TCP connections used

NOTES

Summary

- Applications use client-server paradigm for interaction
- Client
  - Arbitrary application
  - Actively initiates communication
  - Must know server's
    * IP address
    * Protocol port number

Summary (continued)

- Server
  - Specialized program
  - Runs forever
  - Usually offers one service
  - Passively waits for clients
  - Can handle multiple clients simultaneously

NOTES

Summary (continued)

- Socket API
  - Standardized
  - Specifies interface between applications and protocol software
- Socket
  - Operating system abstraction
  - Created dynamically
  - Used by clients and servers

Summary (continued)

- Domain Name System
  - Maps name to IP address
  - Uses on-line servers
  - Uses caching for efficiency
- Two e-mail transfer protocols
  - SMTP
  - POP3

NOTES

Summary (continued)

- IP telephony
  - Transmission of voice telephone calls
  - Several sets of proposed standards
- Remote login
  - Remote, interactive use
  - Protocol is *TELNET*
- File transfer
  - Copy of entire file
  - Protocol is FTP

NOTES

PART XVI

Other Topics:

(Web technologies; Middleware; Network Management; Security; Initialization and Configuration)

World Wide Web

- Major application protocol used on the Internet
- Simple interface
- Two concepts
  - Point
  - Click

Web Components

- Browser
- Web server
- Hypermedia links
- Document representation
- Transfer protocol

NOTES

Browser

- Application program
- User's interface to Web
- Becomes Web client to fetch information from Web server
- Displays information for user

Web Server

- Running program
- Stores set of Web documents
- Responds to request from browser by sending copy of document

NOTES

Web Standards

- *HyperText Markup Language (HTML)*
- *Uniform Resource Locator (URL)*
- *HyperText Transfer Protocol (HTTP)*

HyperText Markup Language (HTML)

- ASCII representation
- Tags to give general layout guidelines
- Does not specify exact placement or format
- Allows document to contain
  - Text
  - Graphics
  - Links to other documents

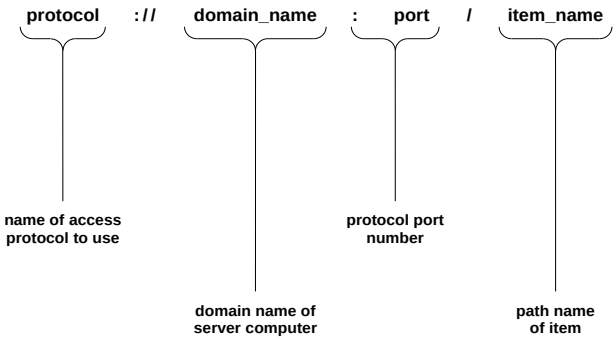
NOTES

Uniform Resource Locator (URL)

- Specifies document on the Web
- Encodes
  - Protocol used to access document
  - Domain name of server
  - Protocol port number of server
  - Path to document

NOTES

General Form Of URL



- Only domain name required
- Defaults
  - Protocol is *http*
  - Port is *80*
  - Path is *index.html*

Use Of Client-Server Paradigm

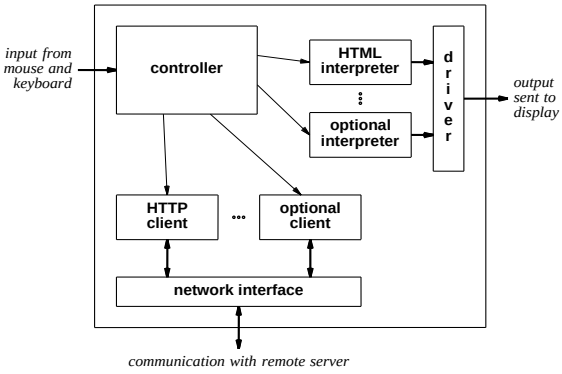
- Web server
  - Makes set of pages available
  - Uses port *80*
- Web client
  - Called a *browser*
  - Creates TCP connection to server
  - Sends requests for items
- Primary protocol known as *HyperText Transfer Protocol HTTP)*

NOTES

Inside A Browser

- Main controller
  - Receives input from user
  - Invokes client and interpreter
- Client
  - One or more built into browser
  - Uses network to fetch items
- Interpreter
  - One or more built in
  - Displays items

Illustration Of A Browser



- Browser contains many components

Alternative Protocol Example

- File transfer service
- Protocol is FTP
- Example URL

  ftp://ftp.cs.purdue.edu/pub/comer/netbook/client.c

- Can be specified in anchor tag

NOTES

Caching In Browsers

- Cache for recently accessed
  - HTML pages
  - Images
- Item normally fetched from cache
- User can override
- HTTP can verify timestamp before fetching new copy

Types Of Web Pages

- Static
  - Stored in file
  - Unchanging
- Dynamic
  - Formed by server
  - Created on demand
  - Output from a program
  - Use *Common Gateway Interface (CGI)* technology

NOTES

Types Of Web Pages
(continued)

- Active
  - Executed at client
  - Consist of a computer program
  - Can interact with user
  - Use *Java* technology

Example Dynamic Document Technology: CGI

- URL specifies
  - Location of Web server
  - CGI program on that server
  - Arguments to program
- Web server
  - Uses TCP for communication
  - Accepts HTTP request from client
  - Runs specified CGI program
  - Returns output to client

NOTES

Example Dynamic Document Technology: CGI (continued)

- CGI program
  - Performs arbitrary computation
  - Often written in a scripting language
  - Produces output file when run
  - Starts output with header

Header In CGI Output

- Stops at first blank line
- Identifies
  - Encoding used
  - Type of document
- Format

  Keyword : information

NOTES

CGI Header Examples

- HTML document header

  ```
  Content Type: text/html
  ```

- Text document header

  ```
  Content Type: text/plain
  ```

- Redirection header

  ```
  Location: /over_here/item4
  ```

Example CGI script

```
#!/bin/sh

#
# CGI script that prints the date and time at which it was run
#

# Output the document header followed by a blank line

echo Content-type: text/plain
echo

# Output the date

echo This document was created on ‘date‘
```

- Generates document
- Document contains three lines of text
  - Header
  - Blank line
  - Document creation date

NOTES

Long-Term State Information

- Program lifetime
  - CGI program invoked by server
  - Program exits after generating output
- To maintain persistent data
  - Write to file on disk
  - Read from file on disk

Example CGI Script With State Information

```
#!/bin/sh
FILE=ipaddrs

echo Content-type: text/plain
echo

# See if IP address of browser's computer appears in our file

if grep -s $REMOTE_ADDR $FILE >/dev/null 2>&1
then

    echo Computer $REMOTE_ADDR has requested this URL previously.

else

    # Append browser's address to the file

    echo $REMOTE_ADDR >> $FILE
    echo This is the first contact from computer $REMOTE_ADDR

fi
```

- Client's IP address in environment variable
- Check if address in file
- Respond to client

NOTES

Encoding Information In A URL

- URL can contain arguments
- Question mark separates CGI path from arguments
- Arguments can encode information

Example Of Arguments Encoding Information

```
#!/bin/sh

echo Content-type: text/html
echo

N=$QUERY_STRING
echo "<HTML>"

case "x$N" in

x)      N=1
        echo "This is the initial page.<BR><BR>"
        ;;

x[0-9]*) N=‘expr $N + 1‘
        echo "You have displayed this page $N times.<BR><BR>"
        ;;

*)      echo "The URL you used is invalid.</HTML>"
        exit 0
        ;;

esac
echo "<A HREF=\"http://$SERVER_NAME$SCRIPT_NAME?$N\">"
echo "Click here to refresh the page.</A> </HTML>"
```

- Argument encodes number of times executed

NOTES

Example Of Script Execution

- Initial document

```
Content-type: text/html

<HTML>
This is the initial page.<BR><BR>
<A HREF="http://www.nonexist.com/cgi/ex4?1">
Click here to refresh the page.</A> </HTML>
```

- Resulting display

This is the initial page.

Click here to refresh the page.

Example Of Script Execution
(continued)

- Generated output

```
Content-type: text/html

<HTML>
You have displayed this page 2 times.<BR><BR>
<A HREF="http://www.nonexist.com/cgi/ex4?2">
Click here to refresh the page.</A> </HTML>
```

- Resulting display

You have displayed this page 2 times.

Click here to refresh the page.

NOTES

Generated URL Values

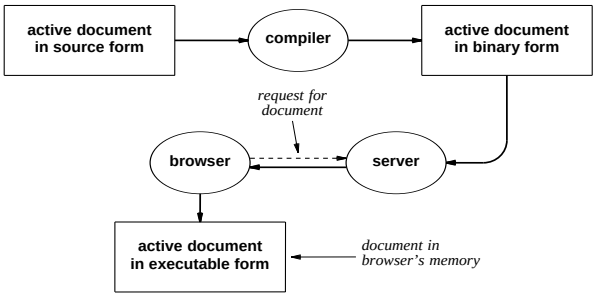
When it generates a document, a dynamic document program can embed state information as arguments in URLs. The argument string is passed to the program for the URL, enabling a program to pass state information from one invocation to the next.

NOTES

Active Document Technology

- Server
  - Sends computer program to client
- Client
  - Runs program locally
- Program
  - Controls display
  - Interacts with user

Active Document Translation



- Compiler produces machine-independent binary
- Browser interprets binary

Java Technology

- Developed by Sun Microsystems
- Used for
  - Conventional applications
  - Active documents (*applets*)
- Includes
  - Programming language
  - Run-time system
  - Class library

NOTES

Java Language Characteristics

- High Level
- General Purpose
- Similar to C++
- Object Oriented
- Dynamic
- Strongly typed
- Statically type checked
- Concurrent

Java Run-Time Environment Characteristics

- Interpretative execution
- Automatic garbage collection
- Multi-threaded execution
- Internet access
- Graphics support

NOTES

Java Library

- Classes for
  - Graphics manipulation
  - Low-level network I/O
  - Interaction with a Web server
  - Run-time system access
  - File I/O
  - Conventional data structures
  - Event capture
  - Exception handling

Choice Of Graphics Interface

Java includes an extensive graphics toolkit that consists of run-time support for graphics as well as interface software. The toolkit allows a programmer to choose a high-level interface, in which the toolkit handles details, or a low-level interface, in which the applet handles details.

NOTES

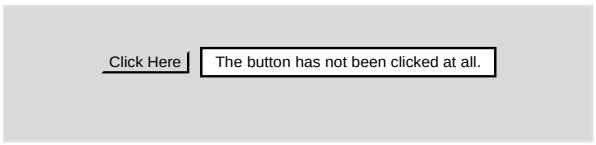
Example Java Applet

- Window with two items
  - Text area
  - Button
- Change text when button clicked

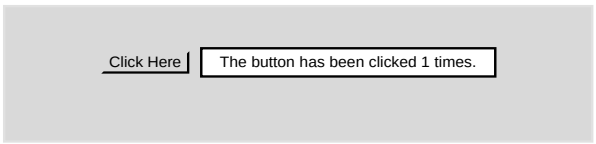
NOTES

Illustration Of Applet Display

- Initial



- After user clicks button



Example Applet Code

```
import java.applet.*;
import java.awt.*;

public class clickcount extends Applet {
  int count;
  TextField f;

  public void init() {
    count = 0;
    add(new Button("Click Here"));
    f = new TextField("The button has not been clicked at all.");
    f.setEditable(false);
    add(f);
  }

  public boolean action(Event e, Object arg) {
    if (((Button) e.target).getLabel() == "Click Here") {
       count += 1;
       f.setText("The button has been clicked " + count + " times.");
    }
    return true;
  }
}
```

Applet Invocation

- Available in HTML
- Uses *applet* tag
- Specifies
  - *Codebase* (machine and path)
  - *Code* (specific class to run)
- Example

```
<applet codebase="www.nonexist.com/pth"
        code="bbb.class">
```


NOTES

Java Functionality

- HTML interface
  - Controls display
  - Interacts with user
- HTTP interface
  - Accesses remote Web documents
  - Invokes other applets
- Exceptions
  - Indicate unanticipated circumstances
  - Can be caught and handled

Middleware

- Tools to help programmers
- Makes client-server programming
  - Easier
  - Faster
- Makes resulting software
  - Less error-prone
  - More reliable

NOTES

Middleware Approach

- Allows programmer to work with familiar language constructs
- Provides tools to help programmer
  - Special translators
  - Libraries
- Automatically generates code for
  - Network communication
  - Connection management

Remote Procedure Call

- Uses standard procedure call paradigm
- Divides program along procedure call boundaries
  - Main program and procedures for user interaction in client side
  - Other procedures in server side

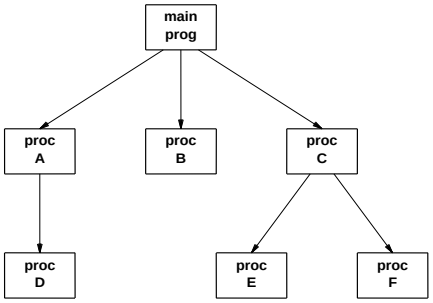
NOTES

Reason For Remote Procedure Call

If a programmer follows the same paradigm used to build conventional programs when building client and server software, the programmer will find the task easier and will make fewer mistakes.

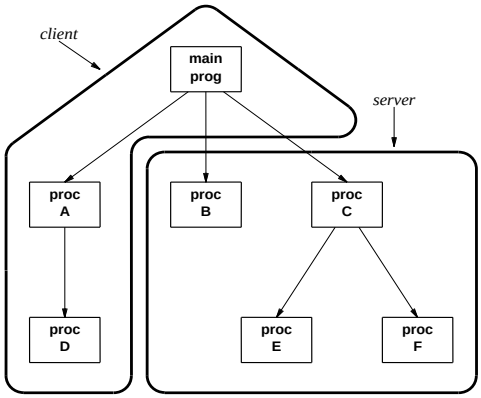
NOTES

Illustration Of Conventional Procedure Call Graph



- Arrow denotes procedure call

Procedure Call Graph Divided Into Client And Server



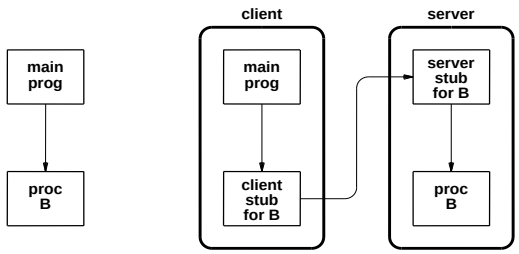
- Division occurs on call boundary
- Main program in client piece

NOTES

Communication Stubs

- Inserted to enable remote call
- Automatically generated
- Use original call interface
- Allow calling and called procedure to remain unchanged

Illustration Of Client And Server Stubs



(a) *(b)*

- Original call in (a)
- Same interface with stubs in (b)

Creating Stubs

- Programmer writes
  - Code for a program
  - Specification of procedure interfaces using *Interface Definition Language (IDL)*
- Middleware generates
  - Client and server stub code
  - Necessary socket calls
  - Data translation

NOTES

Data Representation

- Network can connect heterogeneous computers
- Two computers may use different
  - Integer representations
  - Character codes
  - Floating point representations
- Translation required

Possible Data Translation Schemes

- Use receiver's representation
  - Sender translates all outgoing data
- Use sender's representation
  - Receiver translates all incoming data
- Use external representation (popular)
  - Sender translates to external form before sending
  - Receiver translates from external form after reception

NOTES

Object-Oriented Middleware

- Designed for use with object-oriented programming languages
- Same general scheme as RPC
  - Interface Definition Language
  - Tool to build stubs
  - Libraries to handle network communication
- Uses method invocation instead of procedure call

Network Management

- Used to control and monitor
- Follows client-server model
- Protocol is *Simple Network Management Protocol* (*SNMP*)

NOTES

SNMP Terminology

- Agent
  - Server
  - Runs on network system (e.g., router)
  - Accepts communication from manager station
- Manager station
  - Client
  - Can contact / control multiple agents

SNMP Operations

- Uses fetch-store paradigm
- Each item given name
- Fetch operation used to obtain current value
- Store operation used to control device
- Example: IP routing table
  - Fetch used to obtain route
  - Store used to change route

NOTES

SNMP Variable Names

- Not defined by SNMP
- Separate standard known as *Management Information Base* (*MIB*)
- Names use ASN.1 encoding

Security

- Increasingly important
- Aspects
  - Data integrity
  - Data availability
  - Data confidentiality
  - Privacy

NOTES

Mechanisms

- Message Authentication Code (MAC)
- Passwords
- Digital signatures
- Encryption
- Perimeter security

Perimeter Security

- Placed at connection between organization and rest of Internet
- Implements security policy
  - Restricts arbitrary packets from entering or leaving
- Called *Internet firewall*

NOTES

Firewall Implementation

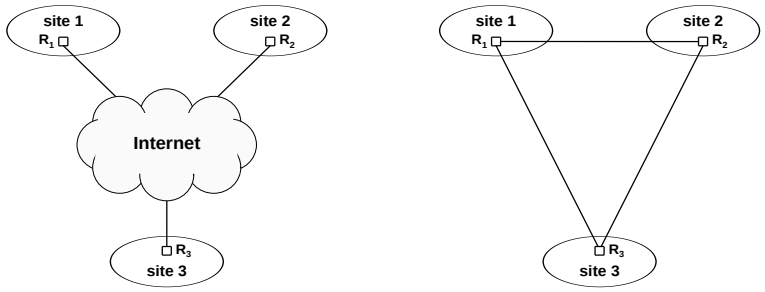
- *Packet Filter*
  - Configurable
  - Specifies which packets can pass
  - Allows manager to specify addresses, protocol ports, and packet types
- Often part of router
- Note: two packet filters and intermediate computer are required for optimal firewall

Virtual Private Network

- Goal: combine advantages
  - Low-cost of conventional Internet connection
  - Confidentiality of leased data circuit
- Uses Internet to transport datagrams between sites
- Encrypts data to prevent snooping
- Uses IP-in-IP encapsulation

NOTES

Illustration Of VPN

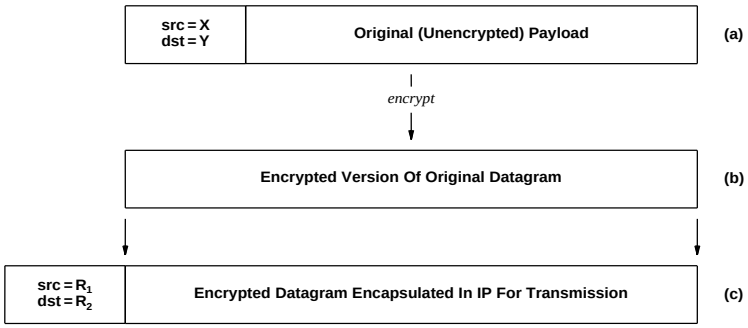


(a) A VPN created by using encryption and route restriction

(b) An equivalent private network created from leased circuits

NOTES

Illustration Of IP-in-IP



- Datagram passes across Internet from router at one site to router at another
- Original source and destination addresses are encrypted

Initialization

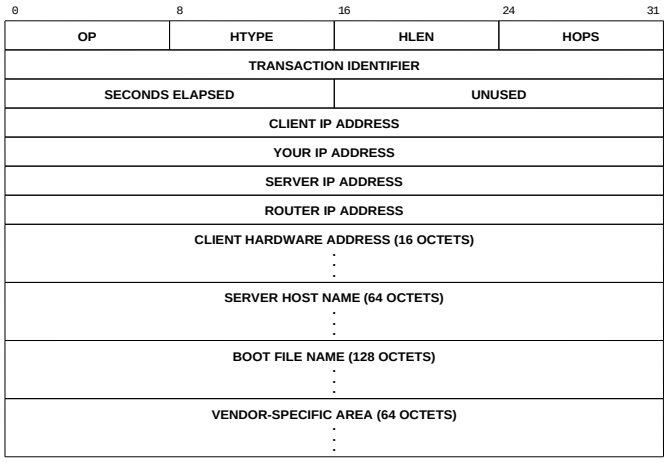
- Configure protocol stack at startup
- Example parameters
  - Local IP address(es)
  - Default IP router address
  - Address mask(s)
  - DNS server address
  - Print server address

NOTES

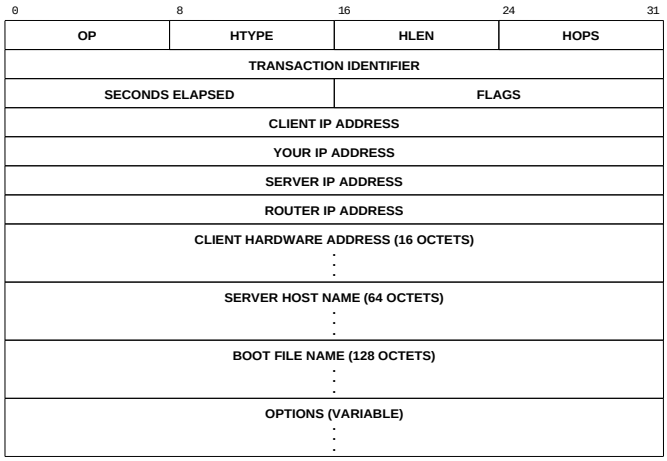
Initialization (Continued)

- Use network to obtain information
  - Broadcast request
  - Receive response from *configuration server*
- Two protocols
  - *BOOTstrap Protocol* (*BOOTP*)
  - *Dynamic Host Configuration Protocol* (*DHCP*)
- Note: response delivered using MAC address because computer does not yet have an IP address

BOOTP Packet Format



DHCP Packet Format



- Allows IP address to be assigned dynamically

NOTES

Summary

- Web is major application in Internet
- Web client called *browser*
- Web server stores documents
- HTML
  - Standard representation
  - Uses tags for markup

Summary (continued)

- HTTP
  - Standard protocol used to fetch document from server
- URL
  - Reference to Web document
  - Encodes
    * Protocol
    * Domain name of server
    * Protocol port number
    * Path of item
    * Only domain name is required

NOTES

Summary (continued)

- Three document types
  - Static (unchanging)
  - Dynamic (generated by server)
  - Active (program runs in browser)
- Example dynamic Web page technology: CGI
- Example active Web page technology: Java

Note: some vendors use *active* to refer to server-side scripting

Summary
(continued)

- Middleware
  - Tools to help build client and server
  - Generates communication stubs automatically
- Network Management
  - Allows manager to control or monitor device
  - Protocol is SNMP

NOTES

Summary
(continued)

- Security
  - Becoming more important
  - Mechanisms include firewalls and encryption
- Initialization
  - Configures protocol stack during startup
  - Two protocols: BOOTP and DHCP
  - DHCP allows dynamic address assignment

NOTES

CONSIDERING DROPPING?

Important Note:

The professor did *not* choose the class time.

Reasons You Should Stay In

- Satisfy your intellectual curiosity
- Understand the network revolution
- Become a better programmer
- Satisfy degree requirements
- Help keep the course going
- It's better than numerical analysis or proving theorems
- Higher salary offers
- All those other students can't be wrong
- It's great fun

NOTES

Reasons You Should Not Stay In

- You have no intellectual curiosity
- Reduce classroom overcrowding
- Stay around for more basketball seasons
- Continue receiving support from your parents
- Postpone real work and big tax bills
- Help inflate the ego of everyone who does stay
- Be around lots of other students
- All those other courses deserve students

Reasons You Should Take This Course From Comer

- He has a record of high teaching evaluations
- He's an expert
- He wrote the textbook
- He knows what industry wants
- He is the only professor teaching it

NOTES

Reasons You Should NOT Take This Course From Comer

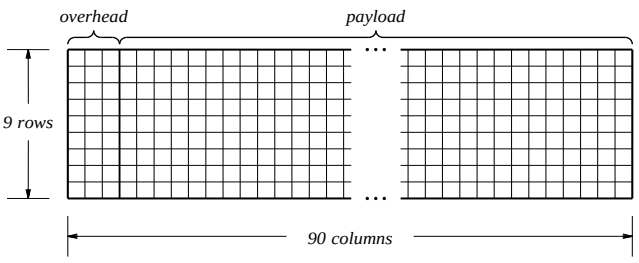
- He has a record of high teaching evaluations
- He's an expert
- He wrote the textbook
- He knows what industry wants
- He expects you to:
  - Attend class
  - Read the text and learn the material
  - Write computer programs
- He does not accept late assignments

SONET

- Stands for Synchronous Optical NETwork
- Digital transmission technology
- Developed by phone companies
- High capacity
- Intended for multiplexing digital voice calls
- Frame size depends on data rate
- Magic constant 125 µsec

NOTES

SONET Frame



- Each frame takes 125 µsec

Snoop Utility

- Captures Ethernet frames
- Applies a filter to select frames
- Stores selected frames in a file on disk

NOTES

Snoop Frame Format (On Disk)

- Multiple frames per file
- File begins with 16-octet header (ignore)
- Each entry in file consists of
  - Snoop header
  - Frame
  - Zero to three octets of padding to next 4-octet boundary
- Snoop header specifies length of frame

Snoop Header

- Found on each entry in file
- 24 octets long
- Specifies length of frame that follows

```
struct   snoophdr {      /* added by snoop         */
long     size;           /* size of frame in bytes */
unsigned char misc[20];  /* skip this              */
}
```


NOTES

Algorithm

- Capture packets in file *F*
- Open file *F* and skip past 16-octet file header
- While file *F* is not empty
  - Read snoop header
  - Extract frame length, *N*
  - Compute *K* as next 4-octet multiple of *N*
  - Read *K* octets into frame buffer
  - Frame is in first *N* octets of buffer