

---

# Distributed Systems

Networking  
And  
Client-server Communication

Paul Krzyzanowski

---

---

## Distributed systems

Independent machines work cooperatively  
without shared memory

They have to talk somehow

Interconnect is the **network**

---

Paul Krzyzanowski • Distributed Systems

---

## Modes of connection

### Circuit-switched

- dedicated path
- guaranteed (fixed) bandwidth
- [almost] constant latency

### Packet-switched

- shared connection
  - data is broken into chunks called packets
  - each packet contains destination address
  - available bandwidth  $\leq$  channel capacity
  - variable latency
- 

Paul Krzyzanowski • Distributed Systems

---

## What's in the data?

For effective communication

- same language, same conventions

For computers:

- electrical encoding of data
  - where is the start of the packet?
  - which bits contain the length?
  - is there a checksum? where is it?  
how is it computed?
  - what is the format of an address?
  - byte ordering
- 

Paul Krzyzanowski • Distributed Systems

---

## Protocols

These instructions and conventions  
are known as **protocols**

---

Paul Krzyzanowski • Distributed Systems

---

## Protocols

Exist at different levels

understand format of address and how to compute checksum

*humans vs. whales  
different wavelengths*

versus

request web page

*French vs. Hungarian*

---

Paul Krzyzanowski • Distributed Systems

## Layering

To ease software development and maximize flexibility:

- Network protocols are generally organized in **layers**
- Replace one layer without replacing surrounding layers
- Higher-level software does not have to know how to format an Ethernet packet  
... or even know that Ethernet is being used

Paul Krzyzanowski • Distributed Systems

## Layering

Most popular model of guiding (not specifying) protocol layers is

### OSI reference model

Adopted and created by ISO

7 layers of protocols

Paul Krzyzanowski • Distributed Systems

## OSI Reference Model: Layer 1

Transmits and receives raw data to communication medium.

Does not care about contents.

voltage levels, speed, connectors



Examples: RS-232, 10BaseT

Paul Krzyzanowski • Distributed Systems

## OSI Reference Model: Layer 2

Detects and corrects errors.

Organizes data into packets before passing it down. Sequences packets (if necessary).

Accepts acknowledgements from receiver.



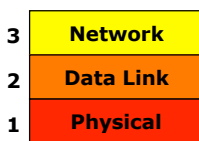
Examples: Ethernet MAC, PPP

Paul Krzyzanowski • Distributed Systems

## OSI Reference Model: Layer 3

Relay and route information to destination.

Manage journey of packets and figure out intermediate hops (if needed).



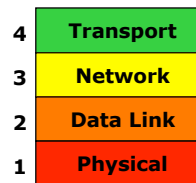
Examples: IP, X.25

Paul Krzyzanowski • Distributed Systems

## OSI Reference Model: Layer 4

Provides a consistent interface for end-to-end (application-to-application) communication. Manages flow control.

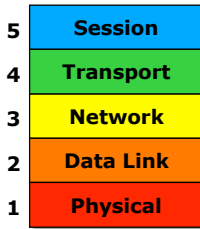
Network interface is similar to a mailbox.



Examples: TCP, UDP

Paul Krzyzanowski • Distributed Systems

## OSI Reference Model: Layer 5



Services to coordinate dialogue and manage data exchange.

Software implemented switch.

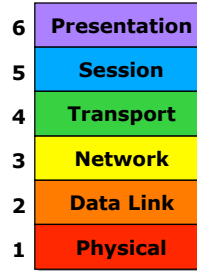
Manage multiple logical connections.

Keep track of who is talking: establish & end communications.

Examples: HTTP 1.1, SSL, NetBIOS

Paul Krzyzanowski • Distributed Systems

## OSI Reference Model: Layer 6



Data representation

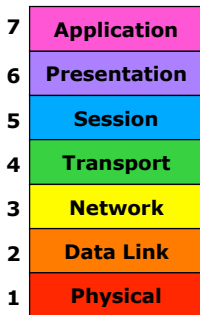
Concerned with the meaning of data bits

Convert between machine representations

Examples: XDR, ASN.1, MIME, MIDI

Paul Krzyzanowski • Distributed Systems

## OSI Reference Model: Layer 7



Collection of application-specific protocols

Examples:  
email (SMTP, POP, IMAP)  
file transfer (FTP)  
directory services (LDAP)

Paul Krzyzanowski • Distributed Systems

## Some networking terminology

## Local Area Network (LAN)

Communications network

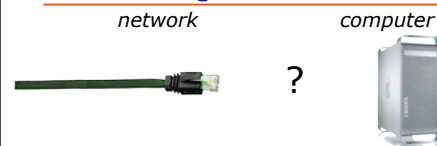
- small area (building, set of buildings)
- same, sometimes shared, transmission medium
- high data rate (often): 1 Mbps – 1 Gbps
- Low latency
- devices are peers
  - any device can initiate a data transfer with any other device

Most elements on a LAN are **workstations**

- endpoints on a LAN are called **nodes**

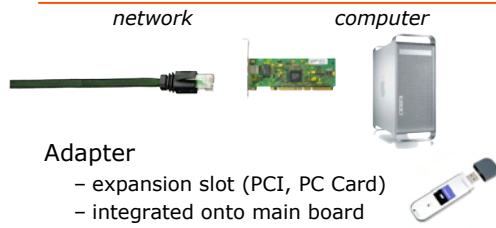
Paul Krzyzanowski • Distributed Systems

## Connecting nodes to LANs



Paul Krzyzanowski • Distributed Systems

## Connecting nodes to LANs



### Adapter

- expansion slot (PCI, PC Card)
- integrated onto main board

Network adapters are referred to as **Network Interface Cards (NICs)**

Paul Krzyzanowski • Distributed Systems

## Media

Wires (or RF, IR) connecting together the devices that make up a LAN

### Twisted pair

- Most common:
  - STP: shielded twisted pair
  - UTP: unshielded twisted pair (e.g. Telephone cable, Ethernet 10BaseT)

### Coaxial cable

- Thin (similar to TV cable)
- Thick (e.g., 10Base5, ThickNet)

### Fiber

### Wireless

Paul Krzyzanowski • Distributed Systems

## Hubs, routers, bridges

### Hub

- Device that acts as a central point for LAN cables
- Take incoming data from one port & send to all other ports

### Switch

- Moves data from input to output port.
- Analyzes packet to determine destination port and makes a virtual connection between the ports.

### Concentrator or repeater

- Regenerates data passing through it

### Bridge

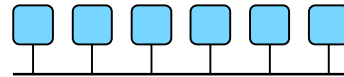
- Connects two LANs or two segments of a LAN
- Connection at data link layer (layer 2)

### Router

- Determines the next network point to which a packet should be forwarded
- Connects different types of local and wide area networks at network layer (layer 3)

Paul Krzyzanowski • Distributed Systems

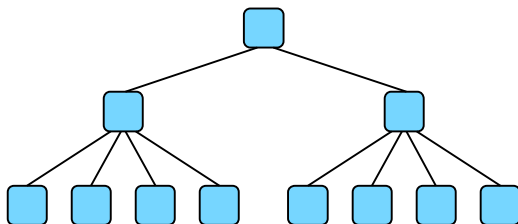
## Networking Topology



**Bus Network**

Paul Krzyzanowski • Distributed Systems

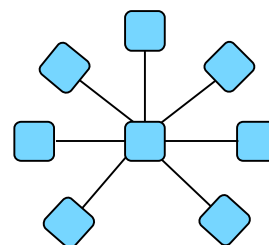
## Networking Topology



**Tree Network**

Paul Krzyzanowski • Distributed Systems

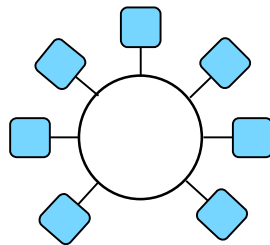
## Networking Topology



**Star Network**

Paul Krzyzanowski • Distributed Systems

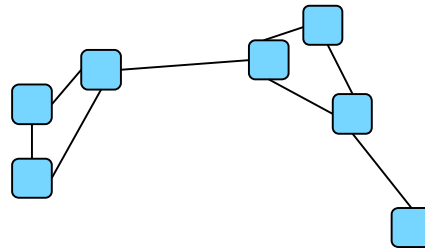
## Networking Topology



**Ring Network**

Paul Krzyzanowski • Distributed Systems

## Networking Topology



**Mesh Network**

Paul Krzyzanowski • Distributed Systems

## Transmission networks

### Baseband

- All nodes share access to network media on an equal basis
- Data uses entire bandwidth of media

### Broadband

- Data takes segment of media by dividing media into channels (frequency bands)

Paul Krzyzanowski • Distributed Systems

## Broadband: RF broadcasts



Paul Krzyzanowski • Distributed Systems

<http://www.ntia.doc.gov/osmhome/allochrt.pdf>

## Broadband/Baseband: Cable TV

### Broadband

55-552 MHz: analog channels 2-78  
553-865 MHz: digital channels 79-136



### Baseband

DOCSIS: Data Over Cable Service Interface Specification (approved by ITU in 1998; DOCSIS 2.0 in 2001)

Downstream: 50-750 MHz range, 6 MHz bandwidth

- up to 38 Mbps
- received by all modems

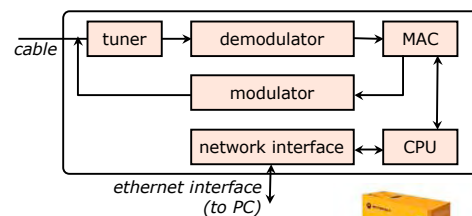
Upstream: 5-42 MHz range

- 30.72 Mbps (10 Mbps in DOCSIS 1.0, 1.1)
- data delivered in timeslots (TDM)

DOCSIS 3.0 will feature **channel bonding** for greater bandwidth

Paul Krzyzanowski • Distributed Systems

## DOCSIS Modem



Restrictions on upload/download rates set by transferring a configuration file to the modem via TFTP when it connects to the provider.



Paul Krzyzanowski • Distributed Systems

## Baseband: Ethernet

Standardized by IEEE as 802.3 standard  
Speeds: 100 Mbps - 1 Gbps typical today

- Ethernet: 10 Mbps
- Fast Ethernet: 100 Mbps
- Gigabit Ethernet: 1 Gbps
- 10 Gbps, 100 Gbps

Network access method is

### Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

- Node first listens to network to see if busy
- Send
- Sense if collision occurred
- Retransmit if collision

Paul Krzyzanowski • Distributed Systems

## Ethernet media

Bus topology (original design)

- originally thick coax (max 500m): 10Base5
- then... thin coax (<200m): 10Base2
  - BNC connector

Star topology (central hub or switch)

- 8 pin RJ-45 connector, UTP cable, 100 meters range
- 10BaseT for 10 Mbps
- 100BaseT for 100 Mbps
- 1000BaseT for 1 Gbps
- Cables
  - CAT-5: unshielded twisted pair
  - CAT-5e: designed for 1 Gbps
  - CAT-6: 23 gauge conductor + separator for handling crosstalk better

Paul Krzyzanowski • Distributed Systems

## Wireless Ethernet media

Wireless (star topology)

- 802.11 (1-2 Mbps)
- 802.11b (11 Mbps - 4-5 Mbps realized)
- 802.11a (54 Mbps - 22-28 Mbps realized)
- 802.11g (54 Mbps - 32 Mbps realized)
- 802.11n (108 Mbps - 30-47 Mbps realized)



Paul Krzyzanowski • Distributed Systems

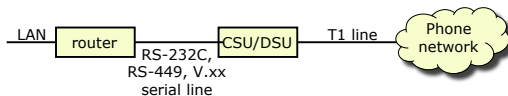
## Connecting to the Internet

- DOCSIS modem via cable TV service
- DSL router
  - Ethernet converted to ATM data stream
  - Up to 20 Mbps up to ~ 2 km.
  - POTS limited to 300-3400 Hz
  - DSL operates > 3500 Hz
- Modem
  - Data modulated over voice spectrum (300-3400 Hz)
  - Serial interface to endpoint
  - V.92: 48 kbps downstream, near 56 kbps up
  - Use PPP or SLIP to bridge IP protocol

Paul Krzyzanowski • Distributed Systems

## Connecting to the Internet

- Dedicated T1 or T3 line
  - T1 line: 1.544 Mbps (24 PCM TDMA speech lines @ 64 kbps)
  - T3 line: 44.736 Mbps (672 channels)
  - CSU/DSU at router presents serial interface
    - Channel Service Unit / Data Service Unit



Paul Krzyzanowski • Distributed Systems

## Connecting to the Internet

- Fiber to the Home, Fiber to the Curb
  - Ethernet interface
  - E.g., Verizon's FiOS 30 Mbps to the home
- Long Reach Ethernet (LRE)
  - Ethernet performance up to 5,000 feet
- Wireless:
  - WiMax
  - EDGE (70-135 Kbps)
  - GPRS (<32 Kbps)

Paul Krzyzanowski • Distributed Systems

## Clients and Servers

---

- send messages to *applications*
  - not just machines
- client must get data to desired *process*
  - server process must get data back to client process
  
- To offer a service, a server must get a **transport address** for a particular service
  - well-defined location

Paul Krzyzanowski • Distributed Systems

## Transport Address

---

Server associates service (process) with transport address before clients communicate with it

machine address → building address  
transport address → apartment number

Before getting access to the service the client must find the transport address

- hard coded
- database (/etc/services, directory server)

Paul Krzyzanowski • Distributed Systems

## Transport provider

---

Layer of software that accepts a network message and sends it to a remote machine

Two categories:

**connection-oriented protocols**

**connectionless protocols**

Paul Krzyzanowski • Distributed Systems

## Connection-oriented Protocols

---

1. establish connection
2. [negotiate protocol]
3. exchange data
4. terminate connection

Paul Krzyzanowski • Distributed Systems

## Connection-oriented Protocols

---

1. establish connection *analogous to phone call*  
*dial phone number*
2. [negotiate protocol] *[decide on a language]*
3. exchange data *speak*
4. terminate connection *hang up*

### virtual circuit service

- provides illusion of having a dedicated circuit
- messages guaranteed to arrive in-order
- application does not have to address each message

vs. **circuit-switched service**

Paul Krzyzanowski • Distributed Systems

## Connectionless Protocols

---

- no call setup
- send/receive data  
(each packet addressed)
- no termination

Paul Krzyzanowski • Distributed Systems

## Connectionless Protocols

*analogous to mailbox*

- no call setup
- send/receive data *drop letter in mailbox*  
(each packet addressed) *(each letter addressed)*
- no termination

### datagram service

- client is not positive whether message arrived at destination
- no state has to be maintained at client or server
- cheaper but less reliable than virtual circuit service

Paul Krzyzanowski • Distributed Systems

## IP – Internet Protocol

Born in 1969 as a research network of 4 machines

Funded by DoD's ARPA

### Goal:

*build an efficient fault-tolerant network that could connect heterogeneous machines and link separately connected networks*

Paul Krzyzanowski • Distributed Systems

## Internet Protocol

- Connectionless protocol designed to handle the interconnection of a large number of local and wide-area networks that comprise the internet
- IP can route from one physical network to another

Paul Krzyzanowski • Distributed Systems

## IP Addressing

Each machine on an IP network is assigned a unique 32-bit number:

- **IP address**, *not* machine address

32 bit addresses → >4 billion addresses!

- routers would need a table of 4 billion entries
- design routing tables so one entry can match multiple addresses
  - hierarchy: addresses physically close will share a common prefix

Paul Krzyzanowski • Distributed Systems

## IP Addressing

cs.rutgers.edu	remus.rutgers.edu
128.6.4.2	128.6.13.3
<u>80 06</u> 04 02	<u>80 06</u> 0D 03

- first 16 bits identify Rutgers
- external routers need only one entry
  - route 128.6.\*.\* to Rutgers

Paul Krzyzanowski • Distributed Systems

## IP Addressing

- IP address
  - **network #**: identifies network machine belongs to
  - **host #**: identifies host on the network
- use network number to route packet to correct network
- use host number to identify specific machine

Paul Krzyzanowski • Distributed Systems

## IP Addressing

Expectation:

- a few big networks and many small ones
- create different **classes** of networks
- use leading bits to identify network

class	leading bits	bits for net #	bits for host
A	0	7 (128)	24 (16M)
B	10	14 (16K)	16 (64K)
C	110	21 (2M)	8 (256)

To allow additional networks within an organization:

- use high bits of host number for a "network within a network" - **subnet**

Paul Krzyzanowski • Distributed Systems

## Classless Inter-Domain Routing

### CIDR

Replace class A, B, C addresses:  
support arbitrary prefix

- Explicitly specify # of bits for network number
- rather than 8 (A), 16 (B), 24 (C) bits

Better match for organizational needs  
machine that needs 500 addresses:

- get a 23-bit network number (512 hosts)
- instead of a class B address (64K hosts)

Paul Krzyzanowski • Distributed Systems

## Classless Inter-Domain Routing

How does a router determine # bits?

CIDR address:

*32-bit-address/bits-for-network-prefix*

- 128.6.13.3/16
- /27 : 1/8 of a class C (32 hosts)
- /24 : class C
- /16 : class B

managing CIDR addresses & prefixes can be a pain

Paul Krzyzanowski • Distributed Systems

## IP addresses

A machine connected to several physical networks will have several IP addresses

- One for each network

Paul Krzyzanowski • Distributed Systems

## IP Special Addresses

- All bits 0
  - Valid only as *source address*
  - "all addresses for this machine"
  - Not valid over network
- All host bits 1
  - Valid only as destination
  - Broadcast to network
- All bits 1
  - Broadcast to all directly connected networks
- Leading bits 1110
  - Class D network
- 127.0.0.0: reserved for local traffic
  - 127.0.0.1 usually assigned to *loopback device*

Paul Krzyzanowski • Distributed Systems

## IPv6 vs. IPv4

IPv4

- 4 byte (32 bit) addresses

IPv6:

- 16-byte (128 bit) addresses
- 4-bit priority field
- Flow label (24-bits)

Paul Krzyzanowski • Distributed Systems

## Names

Human readable names  
e.g. `remus.rutgers.edu`

Hierarchical naming scheme

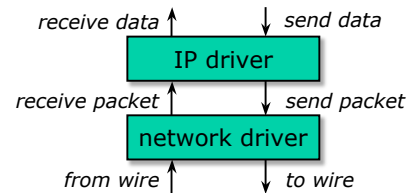
- Look up name in file (`/etc/hosts`)
- Or contact a **name server**
  - which may contact other name servers
  - **Domain Name Service (DNS)**

Paul Krzyzanowski • Distributed Systems

## Getting to the machine

IP is a logical network on top of multiple physical networks

OS support for IP: **IP driver**



Paul Krzyzanowski • Distributed Systems

## IP driver responsibilities

- Get operating parameters from device driver
  - Maximum packet size (MTU)
  - Functions to initialize HW headers
  - Length of HW header
- Routing packets
  - From one physical network to another
- Fragmenting packets
- Send operations from higher-layers
- Receiving data from device driver
- Dropping bad/expired data

Paul Krzyzanowski • Distributed Systems

## Device driver responsibilities

- Controls network interface card
  - Comparable to character driver
- Processes interrupts from network interface
  - Receive packets
  - Send them to IP driver

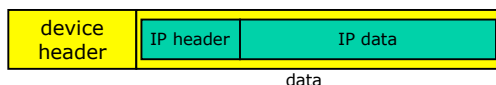
} bottom half
- Get packets from IP driver
  - Send them to hardware
  - Ensure packet goes out without collision

} top half

Paul Krzyzanowski • Distributed Systems

## Network device

- Network card examines packets on wire
  - Compares destination addresses
- Before packet is sent, it must be **enveloped** for the physical network



Paul Krzyzanowski • Distributed Systems

## Device addressing

IP address → ethernet address

**Address Resolution Protocol (ARP)**

1. Check local ARP cache
2. Send broadcast message requesting ethernet address of machine with certain IP address
3. Wait for response (with timeout)

Paul Krzyzanowski • Distributed Systems

## Routing

- Packets take a series of **hops** to get to their destination
  - Figure out the path
- Generate/receive packet at machine
  - check destination
    - If destination = local address, deliver locally
  - else
    - Increment hop count (discard if hop # = TTL)
    - Use destination address to search **routing table**
    - Each entry has address and netmask. Match returns interface
    - Transmit to destination interface
- **Static routing**

Paul Krzyzanowski • Distributed Systems

## Routing

### Dynamic routing

- Class of protocols by which machines can adjust routing tables to benefit from load changes and failures

### Router

- Switching element connects two or more transmission lines (e.g. Ethernet)
- Special-purpose hardware or a general-purpose computer with two or more network interfaces

Paul Krzyzanowski • Distributed Systems

## Transport-layer protocols over IP

- IP sends packets to machine
  - No mechanism for identifying sending or receiving application
- Transport layer uses **port number** to identify application
- TCP – Transmission Control Protocol
- UDP – User Datagram Protocol

Paul Krzyzanowski • Distributed Systems

## TCP – Transmission Control Protocol

- Virtual circuit service (connection-oriented)
- Send acknowledgement for each received packet
- Checksum to validate data
- Data may be transmitted simultaneously in both directions

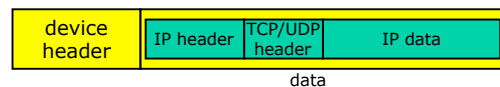
Paul Krzyzanowski • Distributed Systems

## UDP – User Datagram Protocol

- Datagram service (connectionless)
- Data may be lost
- Data may arrive out of sequence
- Checksum for data but no retransmission
  - Bad packets dropped

Paul Krzyzanowski • Distributed Systems

## IP header

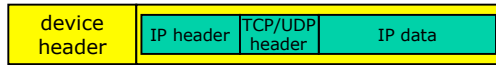


vers	hlen	svc type (tos)	total length
fragment identification		flags	fragment offset
TTL	protocol	header checksum	
source IP address			
destination IP address			
options and pad			

20 bytes

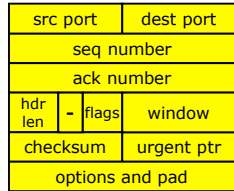
Paul Krzyzanowski • Distributed Systems

## Headers: TCP & UDP



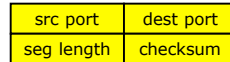
data

TCP header



20 bytes

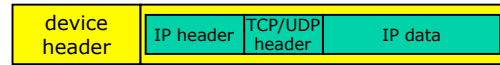
UDP header



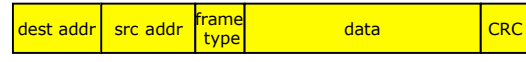
8 bytes

Paul Krzyzanowski • Distributed Systems

## Device header (Ethernet II)



data



6 bytes 6 bytes 2 46-1500 bytes 4

18 bytes + data

Paul Krzyzanowski • Distributed Systems

## Quality of Service Problems in IP

- Too much traffic
  - Congestion
- Inefficient packet transmission
  - 59 bytes to send 1 byte in TCP/IP!
- Unreliable delivery
  - Software to the rescue - TCP/IP
- Unpredictable packet delivery

Paul Krzyzanowski • Distributed Systems

## IP Flow Detection

- Flow detection in routers:
  - Flow: set of packets from one *address:port* to another *address:port* with same protocol
  - Network controls flow rate by dropping or delaying packets
  - With flow detection
    - drop TCP packets over UDP
    - Discard UDP flow to ensure QoS for other flows
- Traffic Shaping
  - Identify traffic flows
  - Queue packets during surges and release later
  - High-bandwidth link to low-bandwidth link
- Traffic Policing
  - Discard traffic that exceeds allotted bandwidth

Paul Krzyzanowski • Distributed Systems

## Dealing with congestion

- FIFO queuing
- Priority queues
- Custom queues
- Flow-based weighted fair queuing
  - Group all packets from a flow together
- Class-based weighted fair queuing
  - Based on protocols, access control lists, interfaces, etc.

Paul Krzyzanowski • Distributed Systems

## Inefficient Packets

- Lots of tiny packets
  - Head-of-line blocking
  - Nagle's algorithm:
    - buffer new data if unacknowledged data outstanding
- Header compression
  - Link-to-link
  - \$ delivery vs. \$ compression

Paul Krzyzanowski • Distributed Systems

## Differentiated Services (soft QoS)

- Some traffic is treated better than others
  - Statistical - no guarantees

Paul Krzyzanowski • Distributed Systems

## TOS bits

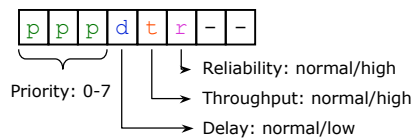
- Advisory tag in IP header for use by routers
- TOS: *Type Of Service*, 4 bits
  - Minimum Delay [0x10]
    - FTP, telnet, ssh
  - Maximum Throughput [0x08]
    - ftp-data, www
  - Maximum reliability [0x04]
    - SNMP, DNS
  - Minimum cost [0x02]
    - NNTP, SMTP

RFC 1349, July, 1992

Paul Krzyzanowski • Distributed Systems

## Differentiated Services (Diff-Serv)

- Revision of interpretation of ToS bits
- ToS field in IP header
  - *Differentiated Services Control Point (DSCP)*



RFC 2475, December 1998

Paul Krzyzanowski • Distributed Systems

## Guaranteed QoS (hard QoS)

Guarantee via end-to-end reservation

Paul Krzyzanowski • Distributed Systems

## Reservation & Delivery Protocols

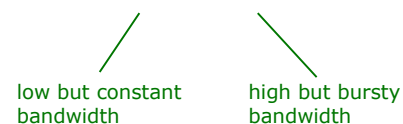
- RSVP: ReSerVation Protocol
  - Hosts request specific quality of service
  - Routers reserve resources
  - RFC 2205
- RTP: Real-Time Transport Protocol
  - *Not* a routing protocol
  - No service guarantees
  - Provides:
    - Payload identification
    - sequence #
    - time stamp

Paul Krzyzanowski • Distributed Systems

## ATM: Asynchronous Transfer Mode

Late 1980's

Goal: Merge voice & data networking



Paul Krzyzanowski • Distributed Systems

## ATM

---

### Traditional voice networking

- Circuit switching
  - Too costly
  - Poor use of resource
  - Does not lend to multicasting

### ATM

- Based on **fixed-size packets** over **virtual circuits**
- Fixed-size cells provide for **predictive scheduling**
- Large cells will not hold up smaller ones
- Rapid switching

Paul Krzyzanowski • Distributed Systems

## ATM

---

### Current standard:

- 53-byte cell: 48-byte data, 5-byte header

### Sender specifies traffic type upon connecting:

<b>CBR</b>	Constant bit-rate	<i>bandwidth</i>	Uncompressed video, voice
<b>VBR</b>	Variable bit-rate	<i>Avg, peak bandwidth</i>	Compressed video, voice
<b>ABR</b>	Available bit-rate	<i>-none-</i>	ftp, web access

Paul Krzyzanowski • Distributed Systems

## ATM

---

### Small cells → lots of interrupts

- >100,000/second

### ATM hardware supports an

#### **ATM Adaptation Layer (AAL)**

- Converts cells to variable-sized (larger) packets:
  - AAL 1: for CBR
  - AAL 2: for VBR
  - AAL 3/4: ABR data
  - AAL 5: ABR data, simplified
  - AAL 6: MPEG-2 video

Paul Krzyzanowski • Distributed Systems

---

## Programming Interfaces

---

## Sockets

---

- IP lets us send data between machines
- TCP & UDP are *transport layer* protocols
  - Contain **port number** to identify transport endpoint (application)
- One popular abstraction for transport layer connectivity: **sockets**
  - Developed at Berkeley

Paul Krzyzanowski • Distributed Systems

## Sockets

---

### Attempt at generalized IPC model

#### Goals:

- communication between processes should not depend on whether they are on the same machine
- efficiency
- compatibility
- support different protocols and naming conventions

Paul Krzyzanowski • Distributed Systems

## Socket

Abstract object from which messages are sent and received

- Looks like a file descriptor
- Application can select particular style of communication
  - Virtual circuit, datagram, message-based, in-order delivery
- Unrelated processes should be able to locate communication endpoints
  - Sockets should be named
  - Name meaningful in the communications domain

Paul Krzyzanowski • Distributed Systems

## Programming with sockets

### Step 1

Create a socket

```
int s = socket(domain, type, protocol)
```

AF\_INET

SOCK\_STREAM  
SOCK\_DGRAM

useful if some families have more than one protocol to support a given service

Paul Krzyzanowski • Distributed Systems

### Step 2

Name the socket (assign address, port)

```
int error = bind(s, addr, addrlen)
```

socket

Address structure  
struct sockaddr\*

length of address structure

Paul Krzyzanowski • Distributed Systems

### Step 3a (server)

Set socket to be able to accept connections

```
int error = listen(s, backlog)
```

socket

queue length for pending connections

Paul Krzyzanowski • Distributed Systems

### Step 3b (server)

Wait for a connection from client

```
int snew = accept(s, clntaddr, &clntalen)
```

socket

new socket for this session

pointer to address structure

length of address structure

Paul Krzyzanowski • Distributed Systems

### Step 3 (client)

---

Connect to server

```
int error = connect(s, svraddr, svraddrlen)
```

socket  
Address structure  
struct sockaddr\*  
length of address structure

---

Paul Krzyzanowski • Distributed Systems

### Step 4

---

Exchange data

Connection-oriented

**read/write**  
**recv/send** (extra flags)

Connectionless

**sendto, sendmsg**  
**recvfrom, recvmsg**

---

Paul Krzyzanowski • Distributed Systems

### Step 5

---

Close connection

**shutdown(s)**

---

Paul Krzyzanowski • Distributed Systems

### Sockets in Java

---

java.net package

Two major classes:

- **Socket**: client-side
- **ServerSocket**: server-side

---

Paul Krzyzanowski • Distributed Systems

### Step 1a (server)

---

Create socket and name it

```
ServerSocket svc =  
    new ServerSocket(port)
```

---

Paul Krzyzanowski • Distributed Systems

### Step 1b (server)

---

Wait for connection from client

```
Server req = svc.accept()
```

↑  
new socket for client session

---

Paul Krzyzanowski • Distributed Systems

## Step 1 (client)

Create socket and name it

```
Socket s = new Socket(address, port);
```

obtained from:  
getLocalHost, getByName,  
or getAllByName

```
Socket s =  
    new Socket("cs.rutgers.edu", 2211);
```

Paul Krzyzanowski • Distributed Systems

## Step 2

Exchange data

obtain InputStream/OutputStream from  
Socket object

```
BufferedReader in =  
    new BufferedReader(  
        new InputStreamReader(  
            s.getInputStream()));  
PrintStream out =  
    new PrintStream(s.getOutputStream());
```

Paul Krzyzanowski • Distributed Systems

## Step 3

Terminate connection

close streams, close socket

```
in.close();  
out.close();  
s.close();
```

Paul Krzyzanowski • Distributed Systems

## Socket Internals

## Protocol Control Block

Client only sends data to {machine, port}

How does the server keep track of  
simultaneous sessions to the same  
{machine, port}?

OS maintains a structure called the  
**Protocol Control Block** (PCB)

Paul Krzyzanowski • Distributed Systems

## Server: svr=socket ()

Create entry in PCB table

Local addr	Local port	Foreign addr	Foreign port	L?	Client

Server	Local addr	Local port	Foreign addr	Foreign port	L?
svr →					

Paul Krzyzanowski • Distributed Systems

### Server: bind (svr)

Assign local port and address to socket  
bind(addr=0.0.0.0, port=1234)

Local addr	Local port	Foreign addr	Foreign port	L?	Client

Server	Local addr	Local port	Foreign addr	Foreign port	L?
svr →	0.0.0.0	1234			

Paul Krzyzanowski • Distributed Systems

### Server: listen (svr, 10)

Set socket for listening

Local addr	Local port	Foreign addr	Foreign port	L?	Client

Server	Local addr	Local port	Foreign addr	Foreign port	L?
svr →	0.0.0.0	1234			*

Paul Krzyzanowski • Distributed Systems

### Server: snew=accept (svr)

Block – wait for connection

Local addr	Local port	Foreign addr	Foreign port	L?	Client

Server	Local addr	Local port	Foreign addr	Foreign port	L?
svr →	0.0.0.0	1234			*

Paul Krzyzanowski • Distributed Systems

### Client: s=socket ()

Create PCB entry

Local addr	Local port	Foreign addr	Foreign port	L?	Client
					← s

Server	Local addr	Local port	Foreign addr	Foreign port	L?
svr →	0.0.0.0	1234			*

Paul Krzyzanowski • Distributed Systems

### Client: s=bind (s)

Assign local port and address to socket  
bind(addr=0.0.0.0, port=7801)

Local addr	Local port	Foreign addr	Foreign port	L?	Client
0.0.0.0	7801				← s

Server	Local addr	Local port	Foreign addr	Foreign port	L?
svr →	0.0.0.0	1234			*

Paul Krzyzanowski • Distributed Systems

### Client: connect (s)

Send *connect* request to server  
[135.250.68.3:7801] to [192.11.35.15:1234]

Local addr	Local port	Foreign addr	Foreign port	L?	Client
0.0.0.0	7801				← s

Server	Local addr	Local port	Foreign addr	Foreign port	L?
svr →	0.0.0.0	1234			*
snew →	192.11.35.15	1234	135.250.68.3	7801	

Paul Krzyzanowski • Distributed Systems

### Client: connect (s)

Server responds with acknowledgement  
[192.11.35.15:1234] to [135.250.68.3 :7801]

Local addr	Local port	Foreign addr	Foreign port	L?	Client
0.0.0.0	7801	192.11.35.15	1234		← S

Server	Local addr	Local port	Foreign addr	Foreign port	L?
svr →	0.0.0.0	1234			*
snew →	192.11.35.15	1234	135.250.68.3	7801	

Paul Krzyzanowski • Distributed Systems

### Communication

Each message from client is tagged as either *data* or *control* (e.g. *connect*)

If data – search through table where FA and FP match incoming message and *listen*=false

If control – search through table where *listen*=true

Server	Local addr	Local port	Foreign addr	Foreign port	L?
svr →	0.0.0.0	1234			*
snew →	192.11.35.15	1234	135.250.68.3	7801	

Paul Krzyzanowski • Distributed Systems

The end