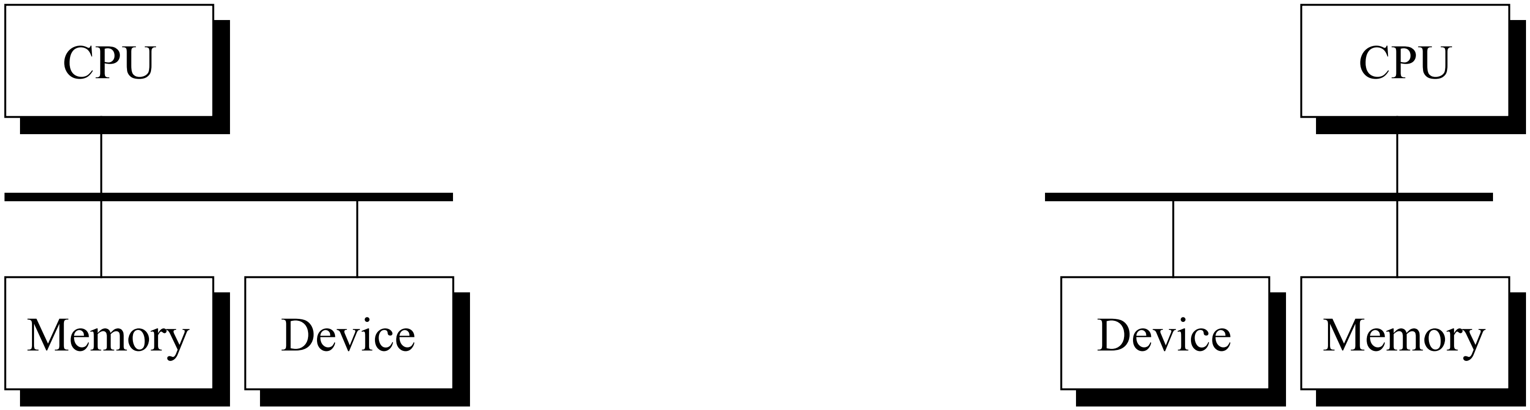
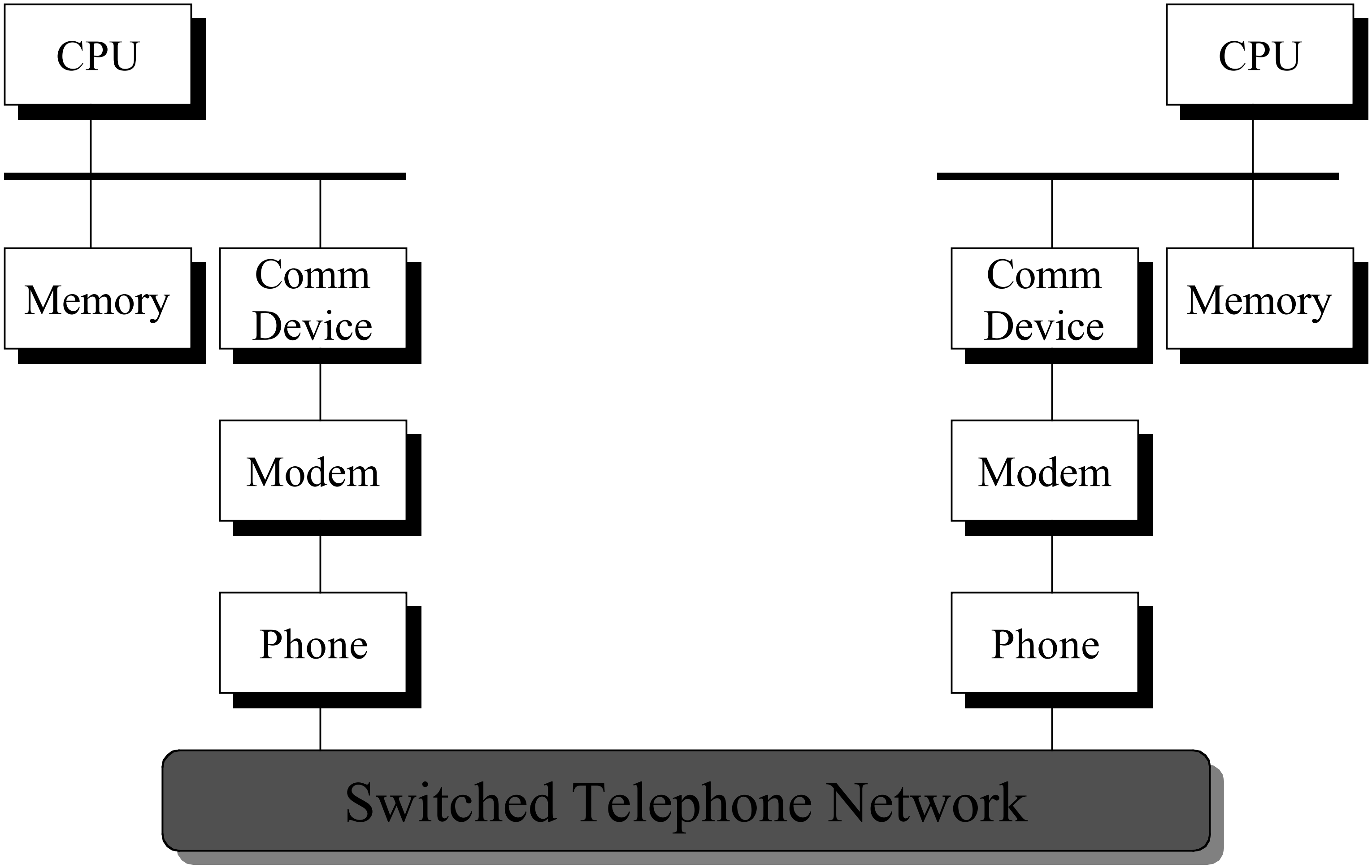
Networks

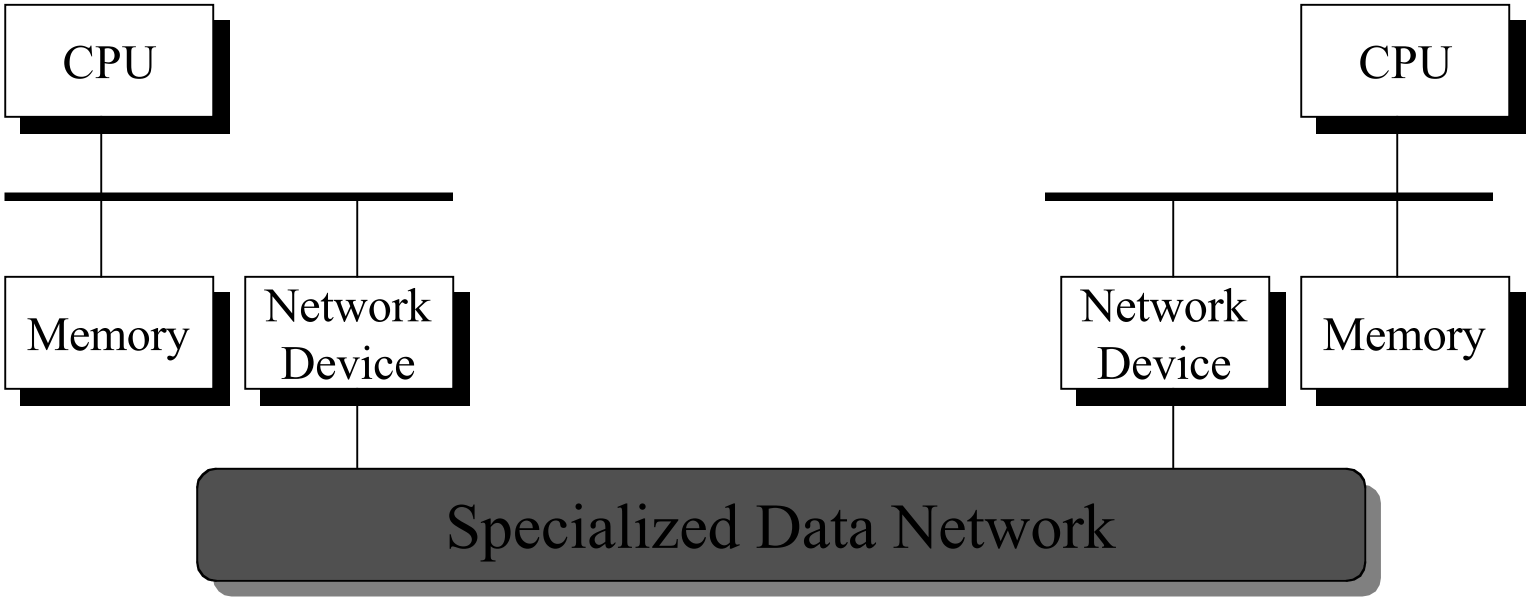
Computer-Computer Comm



Computer-Computer Comm



Data Networks



•WANs, MANs, and LANs
•Specialized communication *protocols*
•Multidrop
•Packet oriented
•Looks like other devices… make it look like a file ...

Multidrop Packet Network

- Need a cost-effective “switch fabric” -- cheaper/better than the telephone network
- To transmit/receive:
    - Sender convert data packet into form suitable for physical transmission
    - Deliver packets to destination host
    - Receiver converts physical signal back into a data packet
- Need a widely-agreed upon set of protocols

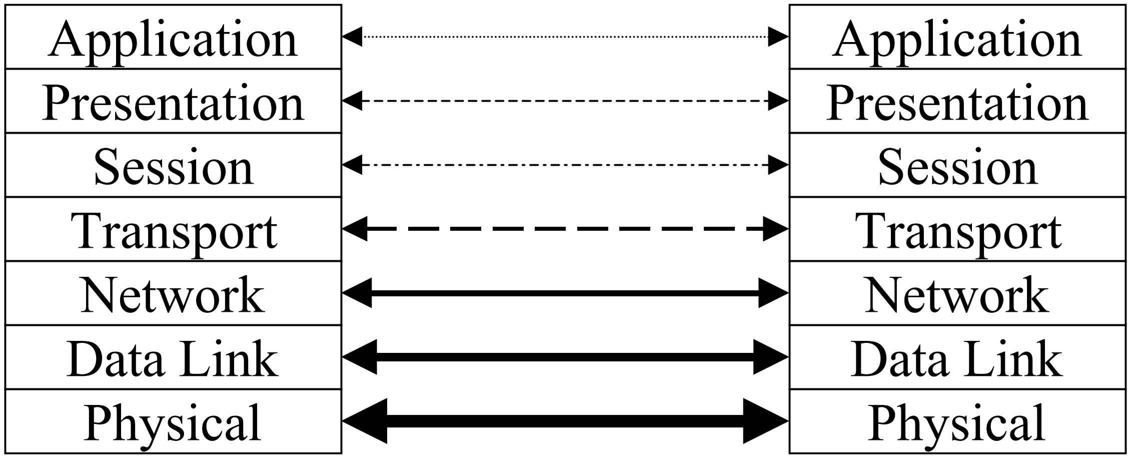
Protocol Tasks

- Control information delivery rates
- Pass info across networks
- Provide fast/reliable IPC-like communication
- Support logical byte streams
- Create other models for communication
  - File transfer
  - Procedure call paradigm
  - Shared memory paradigm
- Translate machine-dependent data representations
- … and more …

Standardizing Protocols

- ANSI X.25
- ARPAnet
- ISO Open Systems Interconnect (OSI) model
  - Now widely used as a reference architecture
  - 7-layer model
  - Provides *framework* for specific protocols (such as IP, TCP, FTP, RPC, RSVP, …)

ISO OSI Model



Application

Presentation

Session

Transport

Network

Data Link

Physical

Application

Presentation

Session

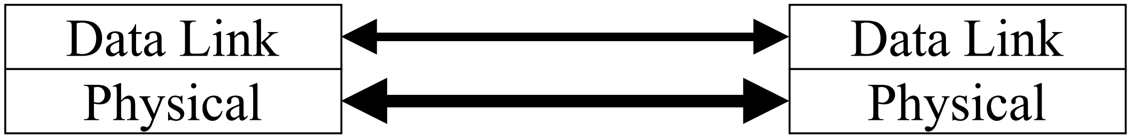
Transport

Network

Data Link

Physical

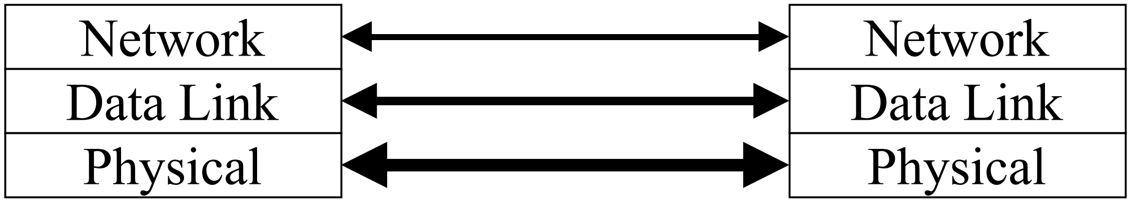
ISO OSI Model



<u>Examples</u>

- Physical/Data Link layer networks: Ethernet, Token Ring, ATM

ISO OSI Model



Network | Network

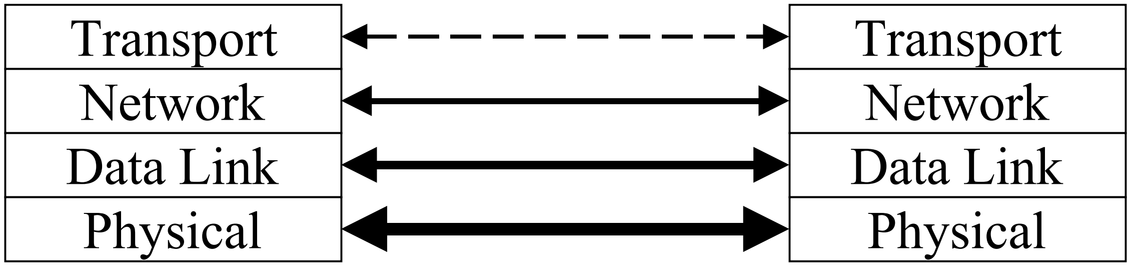
Data Link | Data Link

Physical | Physical

<u>Examples</u>

- Physical/Data Link layer networks: Ethernet, Token Ring, ATM
- Network layer net: The Internet

ISO OSI Model

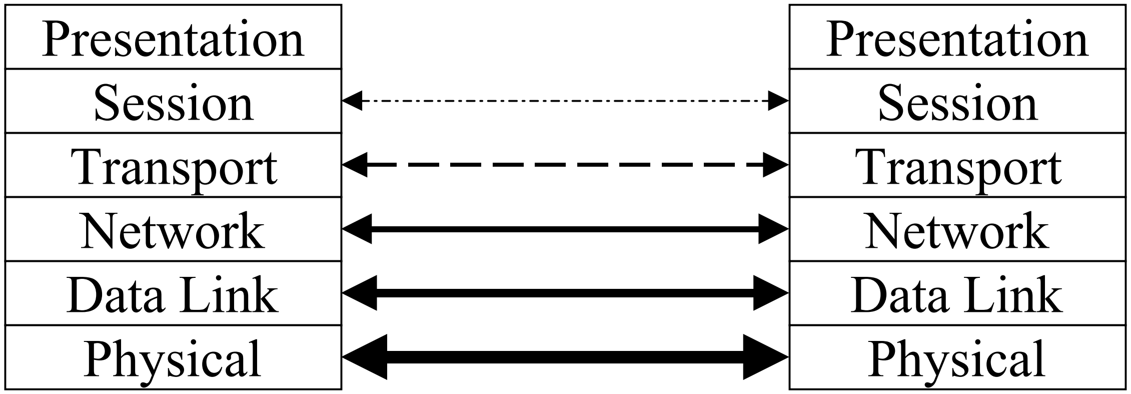


Transport	Transport
Network	Network
Data Link	Data Link
Physical	Physical

<u>Examples</u>

- Physical/Data Link layer networks: Ethernet, Token Ring, ATM
- Network layer net: The Internet
- Transport layer net: TCP-based network

ISO OSI Model

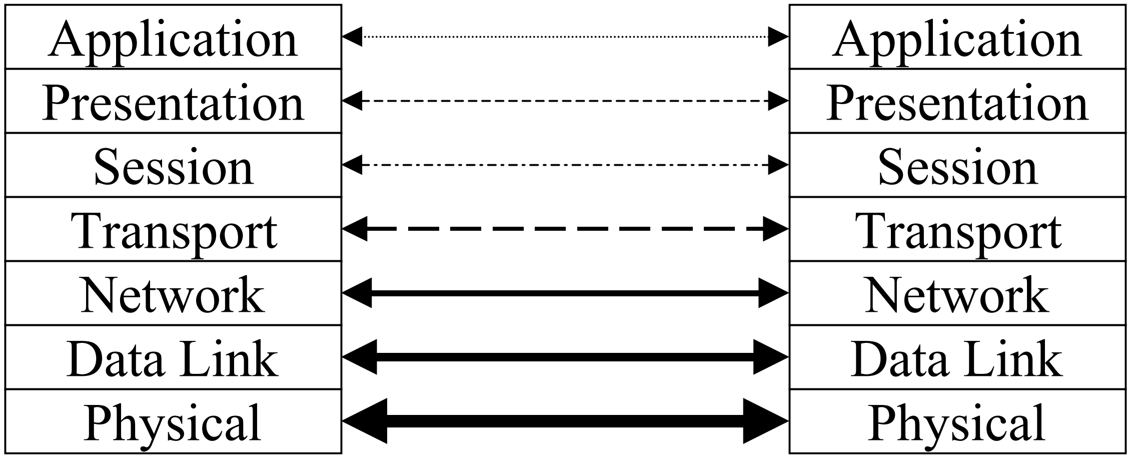


Presentation | Presentation
Session | Session
Transport | Transport
Network | Network
Data Link | Data Link
Physical | Physical

Examples

- Physical/Data Link layer networks: Ethernet, Token Ring, ATM
- Network layer net: The Internet
- Transport layer net: TCP-based network
- Presentation/Session layer net: http/html, RPC, PVM, MPI

ISO OSI Model



Application

Presentation

Session

Transport

Network

Data Link

Physical

Application

Presentation

Session

Transport

Network

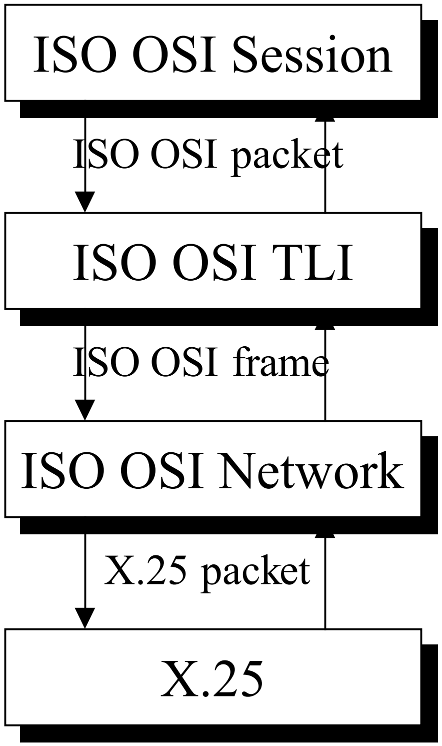
Data Link

Physical

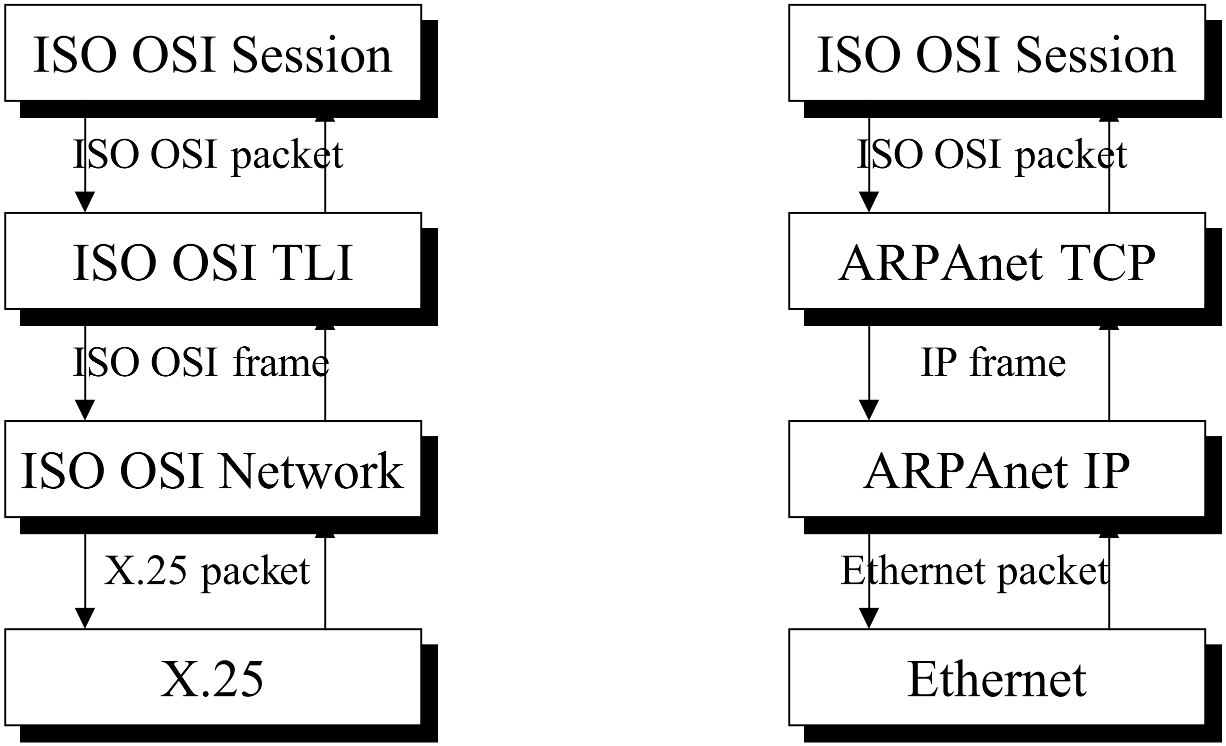
<u>Examples</u>

•Physical/Data Link layer networks: Ethernet, Token Ring, ATM
•Network layer net: The Internet
•Transport layer net: TCP-based network
•Presentation/Session layer net: http/html, RPC, PVM, MPI
•Applications, e.g., WWW, window system, numerical algorithm

ISO OSI & TCP/IP



ISO OSI & TCP/IP

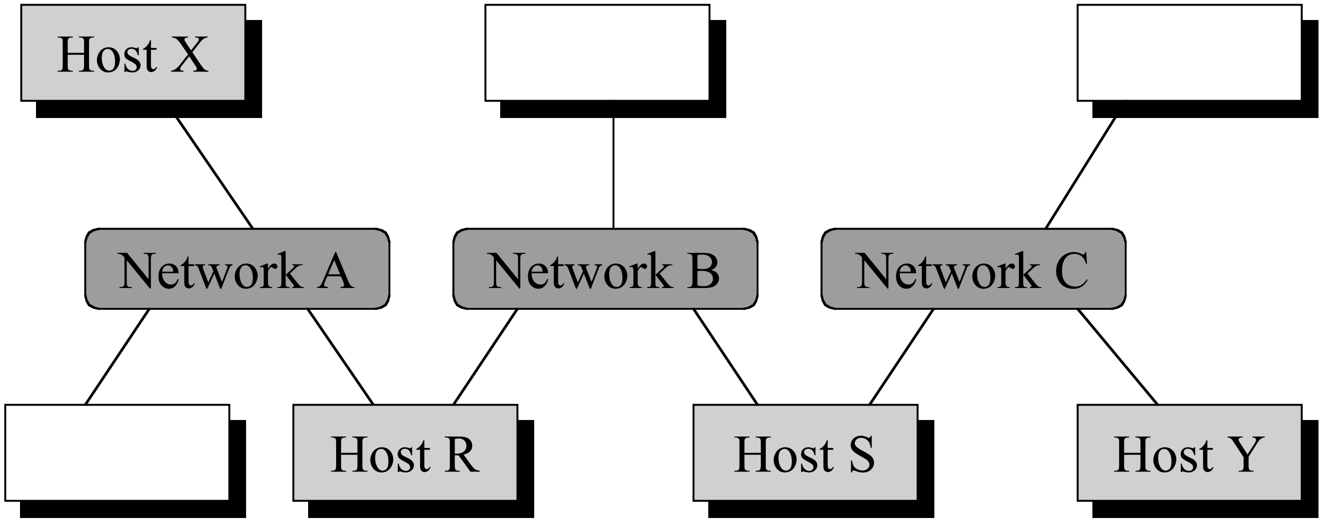


Low Level Protocols

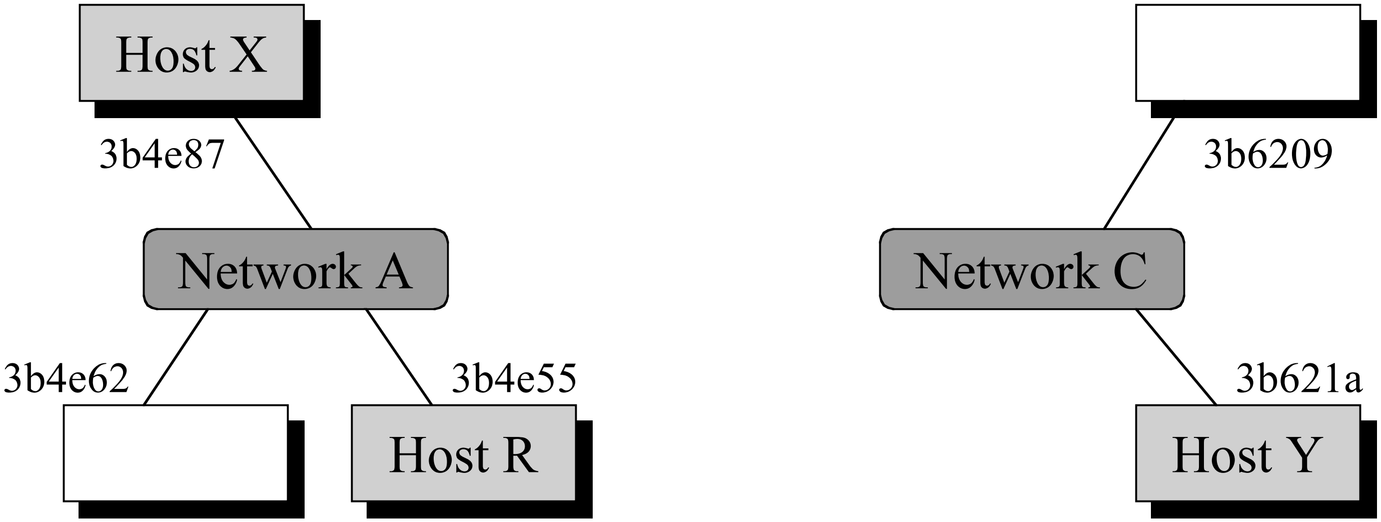
- Physical layer: Signaling technology
- Data link layer: Frame management
- *All done in hardware*
- Examples
  - Ethernet
  - Token ring
  - X.25
  - ATM
- Read pages 463-471

Network Layer

- Primary purpose is to combine networks
- *Internet protocol* (IP) is dominant protocol
- Creates an internet *address space*
- Implements packet *routing* across networks

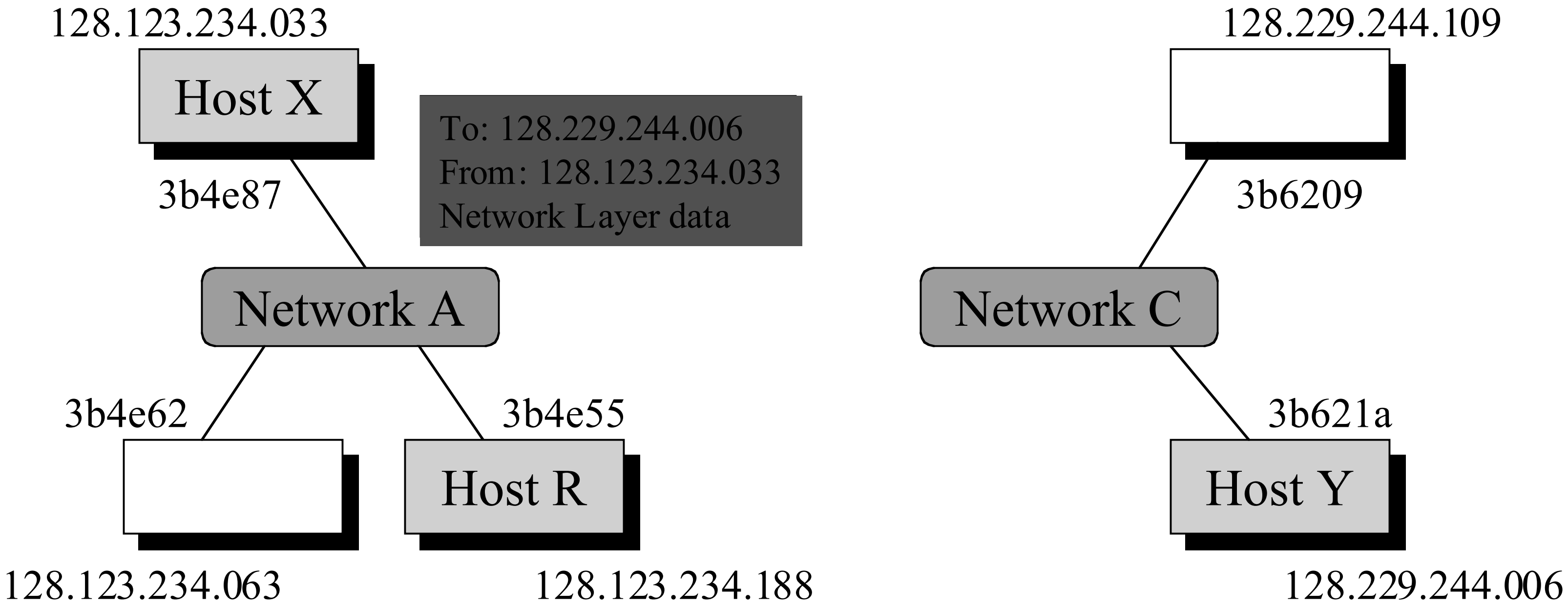


Addressing & Routing



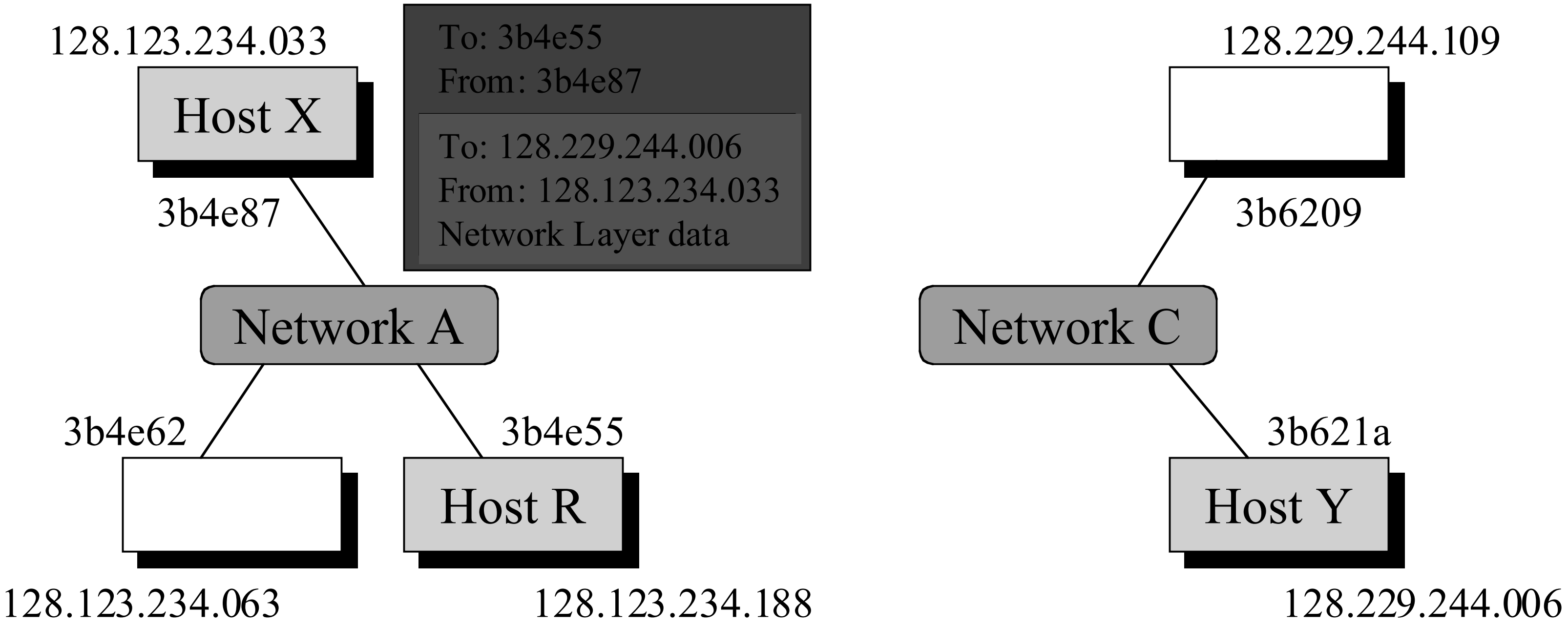
- Host X does not know how to send to Host Y
- Can send a frame to Host R for forwarding
- What should it tell Host R?

Addressing & Routing



- Host X does not know how to send to Host Y
- Can send a frame to Host R for forwarding
- What should it tell Host R?
- Internet address spans all machines

Addressing & Routing

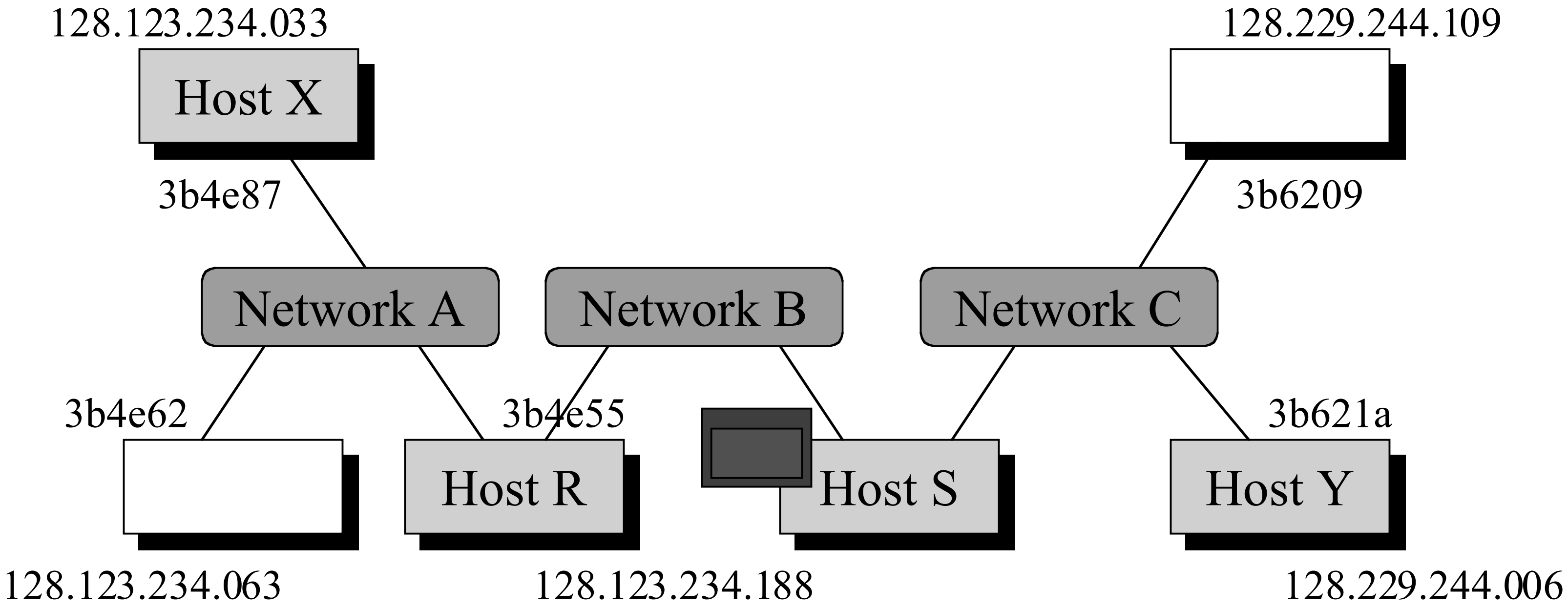


To: 3b4e55
From: 3b4e87

To: 128.229.244.006
From: 128.123.234.033
Network Layer data

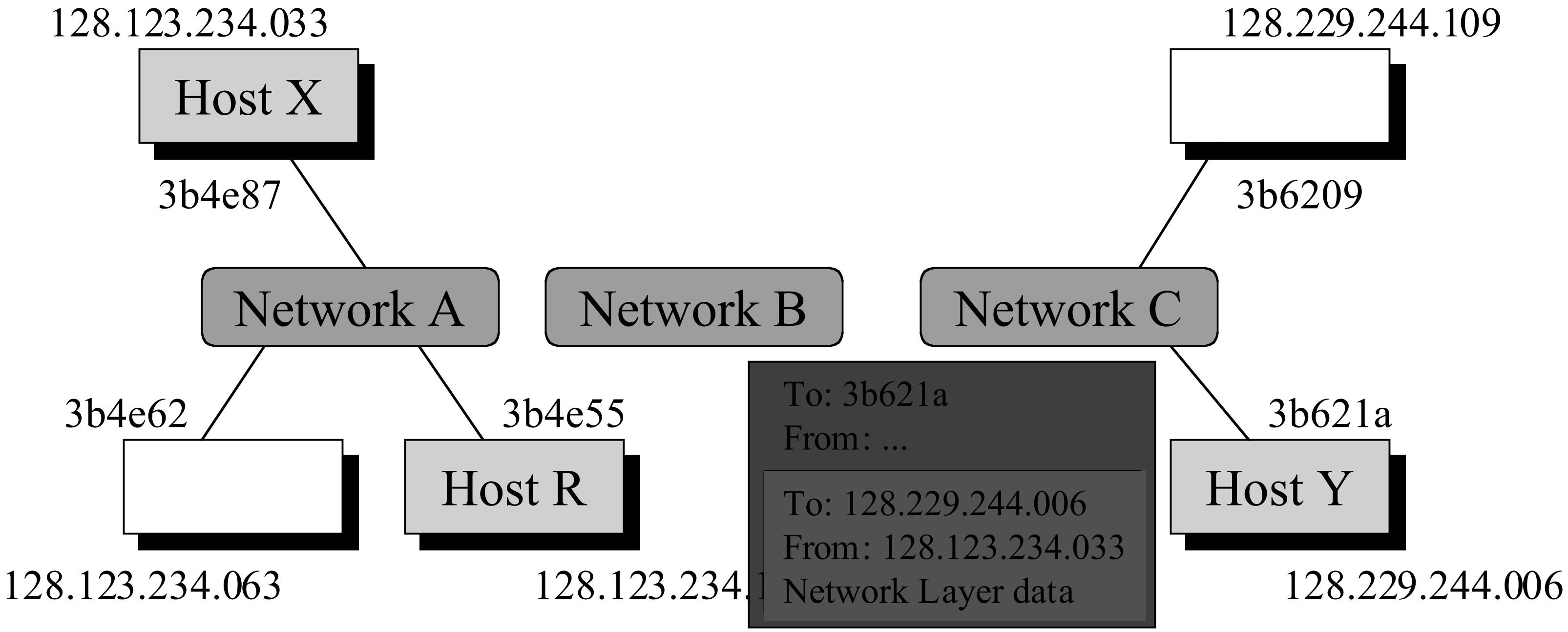
- Host X does not know how to send to Host Y
- Can send a frame to Host R for forwarding
- What should it tell Host R?
- Internet address spans all machines
- Send *encapsulated* packet to Host R with Host Y

Addressing & Routing



- Host X does not know how to send to Host Y
- Can send a frame to Host R for forwarding
- What should it tell Host R?
- Internet address spans all machines
- Send *encapsulated* packet to Host R with Host Y

Addressing & Routing



128.123.234.033

Host X

3b4e87

128.229.244.109

3b6209

Network A

Network B

Network C

3b4e62

3b4e55

3b621a

Host R

Host Y

128.123.234.063

128.123.234.1

128.229.244.006

To: 3b621a
From: ...

To: 128.229.244.006
From: 128.123.234.033
Network Layer data

- Host X does not know how to send to Host Y
- Can send a frame to Host R for forwarding
- What should it tell Host R?
- Internet address spans all machines
- Send *encapsulated* packet to Host R with Host Y
- Data Link frame is received by Host Y

More on the Network Layer

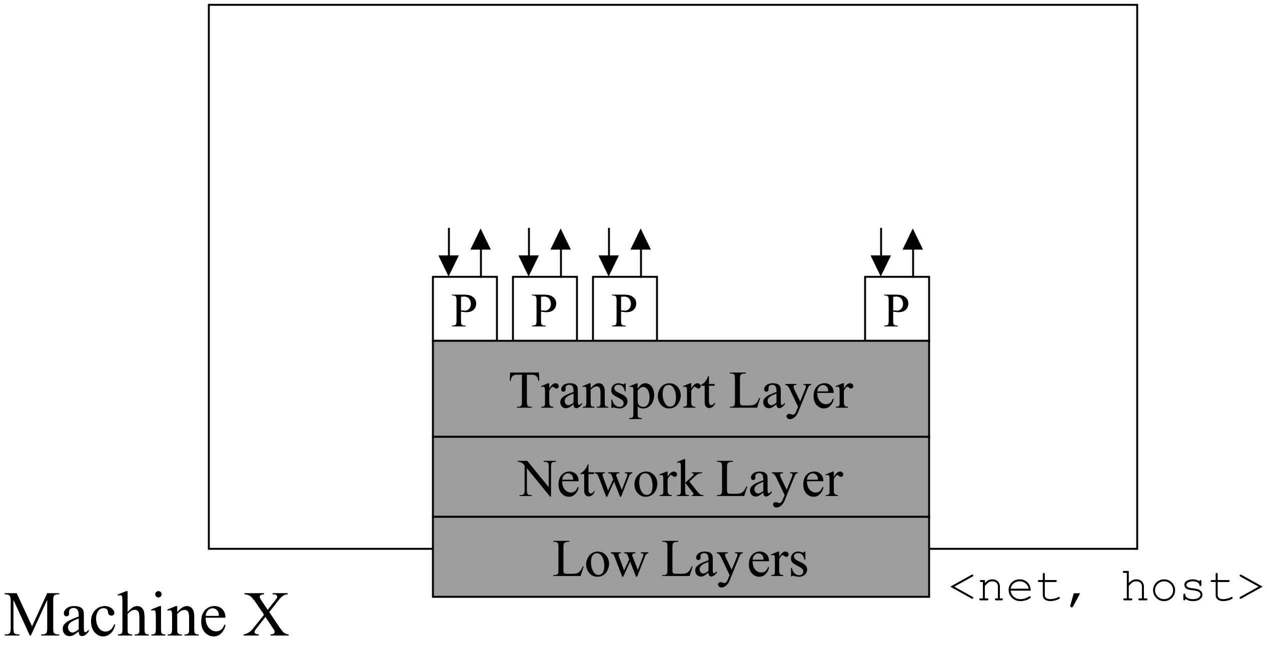
- Implements internet addressing & routing
- ARPAnet IP protocol is dominant -- underlies *the Internet*
- Intermediate hosts are called *gateways*
  - Connected to two or more networks
  - Runs IP routing software
  - `nag` is a gateway for the teaching lab
  - Read pages 471-477

Transport Layer

- Provides yet another address extension
  - IP references onlyu networks and hosts
  - Transport layer adds *ports* -- logical endpoints
  - Address form is `<net, host, port>`
- Two primary protocols (both from ARPAnet)
  - User Datagram Protocol (UDP)
    - User-space interface to IP packets
    - No guarantee that packet will be delivered
  - Transmission Control Protocol (TCP)
    - Provides a stream-oriented interface to the network
    - Reliable delivery

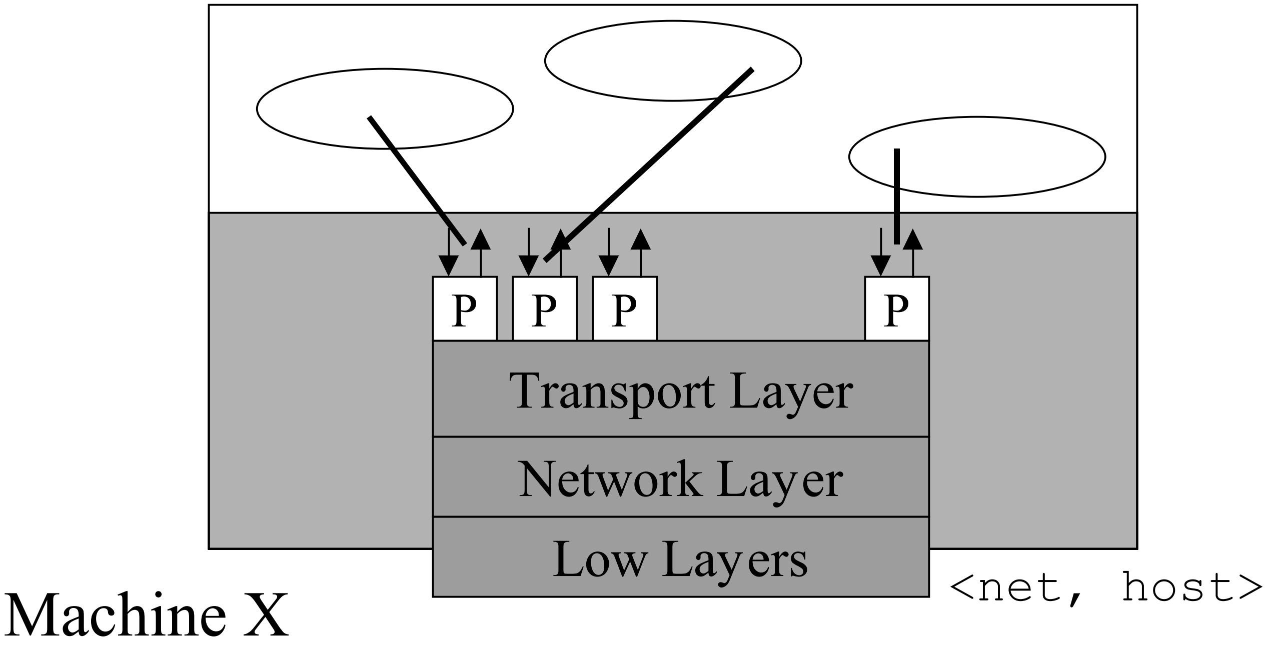
Communication Ports

- Global name for a “mailbox”
- Will be many ports at one <net, host>



Communication Ports

- Global name for a “mailbox”
- Will be many ports at one <net, host>
- Each port can be *bound* to an address



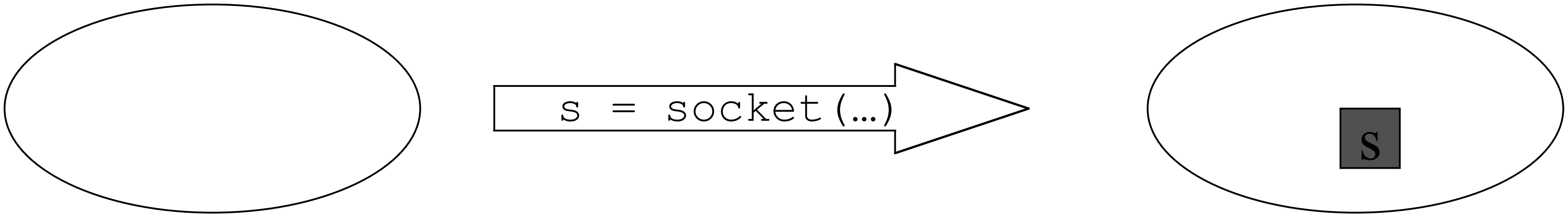
BSD Sockets

- Sockets are comm ports in BSD UNIX
- Semantics resemble pipes (files)
- Bidirectional

BSD Sockets

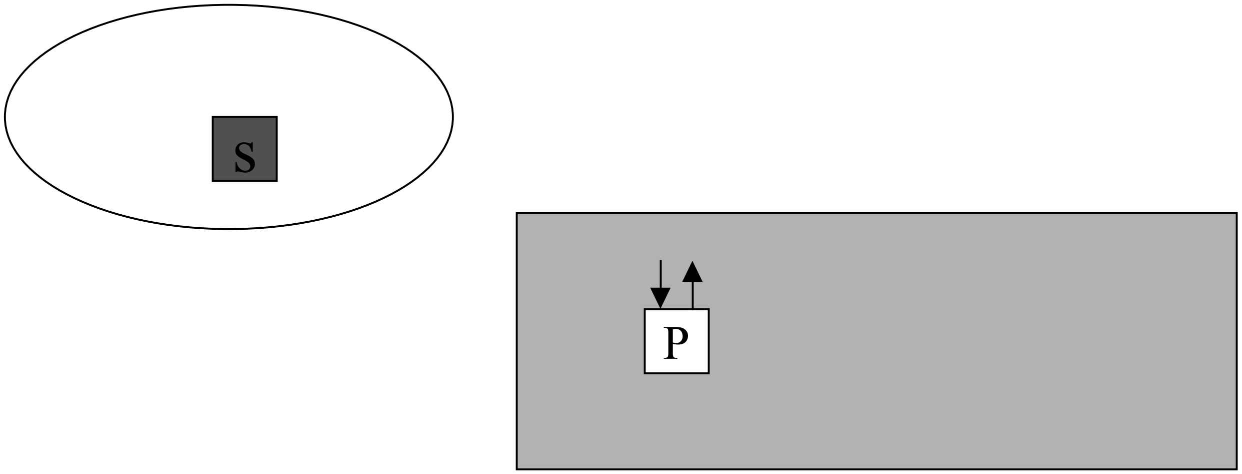
- Sockets are comm ports in BSD UNIX
- Semantics resemble pipes (files)
- Bidirectional

```
int socket(int addressFamily, int socketType, int protocolNo)
```



BSD Sockets (cont)

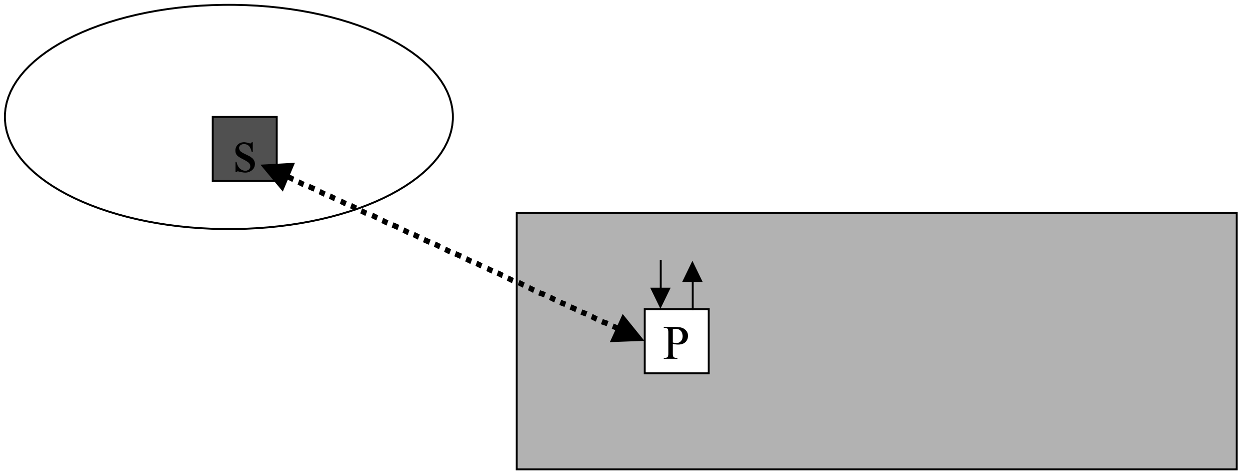
- Once a socket has been created, it can be bound to an internet port



BSD Sockets (cont)

- Once a socket has been created, it can be bound to an internet port

```
int bind(int skt, struct sockadrr *addr, int addrLength)
```



- Example code available on the web page

UDP

- *Datagram* ("*connectionless*") service
  - Similar to disk I/O level of service
- Logically associated with an IP packet & Data Link frame (but not physically)
- Best-effort delivery of datagrams, but:
  - Datagram may be dropped (lost)
  - Datagrams may be delivered out of order
- Efficient, relative to TCP

Using UDP

```
/* Set up a socket to talk to the server */
    skt = socket(AF_INET, SOCK_DGRAM, 0);
    host = gethostbyname(remoteHostName);
    bzero(&remote, sizeof(remote));
    remote.sin_family = host->h_addrtype;
    remote.sin_port = htons(remotePort);
    bcopy(host->h_addr, &remote.sin_addr, host->h_length);
/* Export the socket to a port (and IP address) */
    host = gethostbyname(localHostName);
    bzero(&local, sizeof(local));
    local.sin_family = host->h_addrtype;
    local.sin_port = htons(localPort);
    bcopy(host->h_addr, &local.sin_addr, host->h_length);
    if(bind(skt, &local, sizeof(local))) {
        printf("Bind error ... restart\n");
        exit(1);
    }
    . . .
    sendto(s, outBuf, strlen(outBuf), 0, remote, sizeof(remote));
    if((len = recv(s, inBuf, BUFLEN, 0)) > 0) {. . .}
```

TCP

- *Connected* (or *virtual circuit*) protocol
- Interface allows programmer to read/write a byte stream over the network
- Byte stream is mapped into a series of packets
  - Reliable delivery
  - Each packet must be acknowledged
  - Effectively 2 packets per transmission
- Must open/close a connection before use

Using TCP -- Client

```
skt = socket(AF_INET, SOCK_STREAM, 0);
host = gethostbyname(serverHostName);
bzero(&listener, sizeof(listener));
listener.sin_family = host->h_addrtype;
listener.sin_port = htons(port);
bcopy(host->h_addr, &listener.sin_addr, host->h_length);
if(connect(skt, &listener, sizeof(listener))) {
    printf("Connect error ... restart\n");
    printf("(Must start Server end first)\n");
    exit(1);
};
. . .
write(s, outBuf, BUFLEN);
if((len = read(s, inBuf, BUFLEN)) > 0) {. . .}
```

Using TCP -- Server

```
skt = socket(AF_INET, SOCK_STREAM, 0);  /* Produce an inet address */
host = gethostbyname(serverHostName);
bzero(&listener, sizeof(listener));
listener.sin_family = host->h_addrtype;
listener.sin_port = htons(port);
bcopy(host->h_addr, &listener.sin_addr, host->h_length);
if(bind(skt, &listener, sizeof(listener))) {
    printf("Bind error ... restart\n");
    exit(1);
}
listen(skt, BACKLOG);    /* Listen for a request */
newSkt = accept(skt, &client, &clientLen);
if (fork() == 0) {
    close(skt);     /* Child doesn't need listener socket */
    . . .
}
close(newSkt);          /* Parent doesn't need the new socket */

if((len = read(s, inBuf, BUFLEN)) > 0) { . . .}
write(s, outBuf, BUFLEN);
```

Client-Server Paradigm

- Making a connection in TCP is an example of the *client-server paradigm* for distributed computing
  - Active component is the client
    - Runs autonomously
    - Decides when it wants to use server
  - Passive component is the server
    - Persistent
    - Always waiting for a client to request service
- Not a machine -- just software