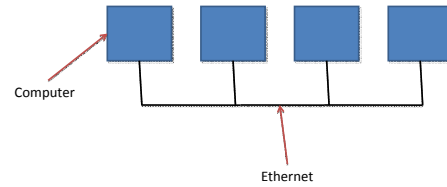


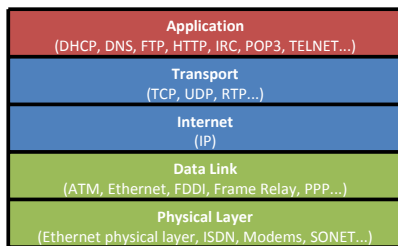
Network Communication

Jonathan Misurda
jmisurda@cs.pitt.edu

Network



Internet Layer Model



Internet Protocol (IP)

- **Protocol** – A standard procedure for regulating data transmission between computers.
- **IP Addresses** – 32-bit (v4) or 128-bit (v6) number denoting an destination or source
- **Datagram** – (play on telegram) A message with no acknowledgement

Transmission Control Protocol (TCP)

- **Connection-oriented** – Make a circuit with a remote machine
- **Port** – a number representing a particular listener on a machine
- Guarantees data arrives, and in-order

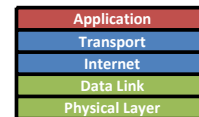
User Datagram Protocol (UDP)

- **Connectionless** – Send and forget
- No guarantee data arrives or is in the order sent

IP Packet

	Bits 0-3	4-7	8-15	16-18	19-31
0	Version	Header length	Type of Service	Total Length	
32	Identification			Flags	Fragment Offset
64	Time to Live		Protocol	Header Checksum	
96	Source Address				
128	Destination Address				
160	Options				
192-	Data				

Packets



BERKLEY SOCKETS

Socket

- UNIX treats everything as a file
 - File Descriptor
 - read()/ write()
- Treat network as a file called a socket
- Berkley sockets are de facto standard API

socket()

- Creates a Socket Descriptor

```
#include <sys/types.h>
```

```
#include <sys/socket.h>
```

```
int socket(int domain, int type, int protocol);
```

Parameter	Values
domain	<ul style="list-style-type: none"> •PF_INET for IPv4 •PF_INET6 for IPv6
type	<ul style="list-style-type: none"> •SOCK_STREAM •SOCK_DGRAM •SOCK_SEQPACKET •SOCK_RAW
protocol	IPPROTO_IP (defined as 0)

SERVER STUFF

bind()

- Attach a socket to a port

```
int bind(int sockfd, struct sockaddr *addr, int
  addrlen);
```

```
memset(&my_addr, 0, sizeof(struct sockaddr));
addr.sin_family = AF_INET;
addr.sin_port = htons(PORT);
addr.sin_addr.s_addr = net_addr("127.0.0.1");
```

listen()

- Set up a listening socket

```
int listen(int sockfd, int backlog);
```

backlog – how many pending connections are allowed to wait (OS max usually around 20, set to lower, ~10)

accept()

- Block and wait for connection to occur

```
int accept(int sockfd, struct sockaddr
  *cliaddr, socklen_t *addrlen);
```

- Will return information about the client through the structure (can be NULL)

send() and recv()

```
int send(int sockfd, const void *msg, int len,
  int flags);
```

```
int recv(int sockfd, void *buf, int len, int
  flags);
```