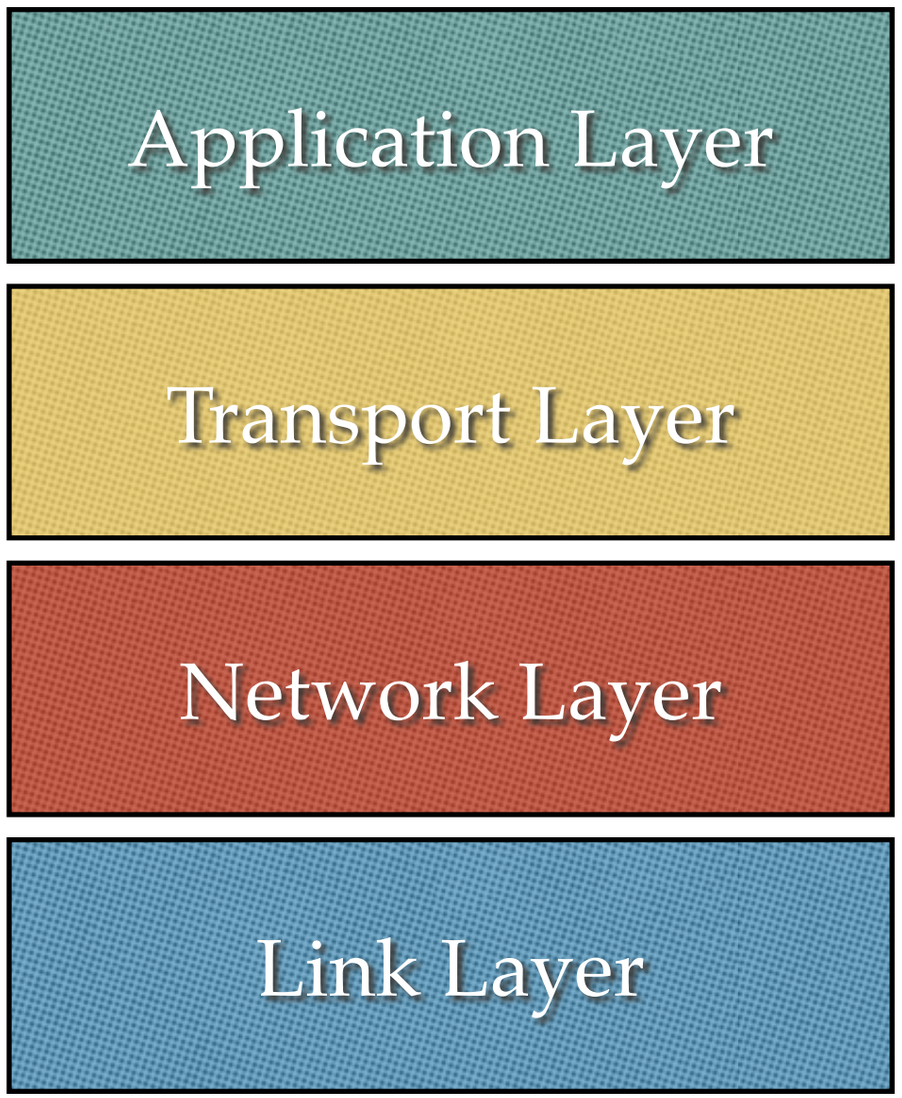
Network Implementation

John Criswell
University of Rochester

Networking Overview

Networking Layers



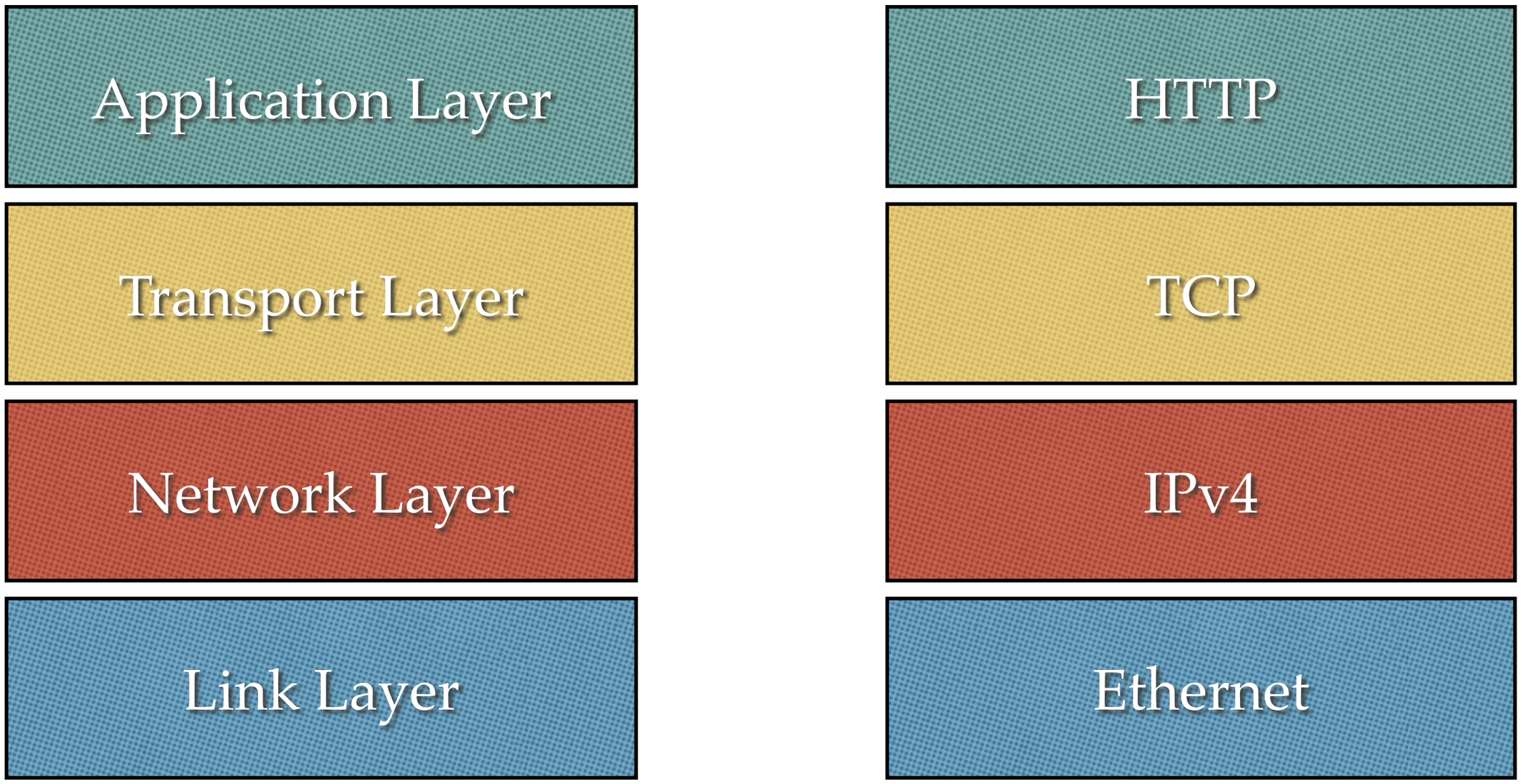
Format of Application Data

Which application?

Which machine on the Internet?

Which machine on the local network?

Networking Layers



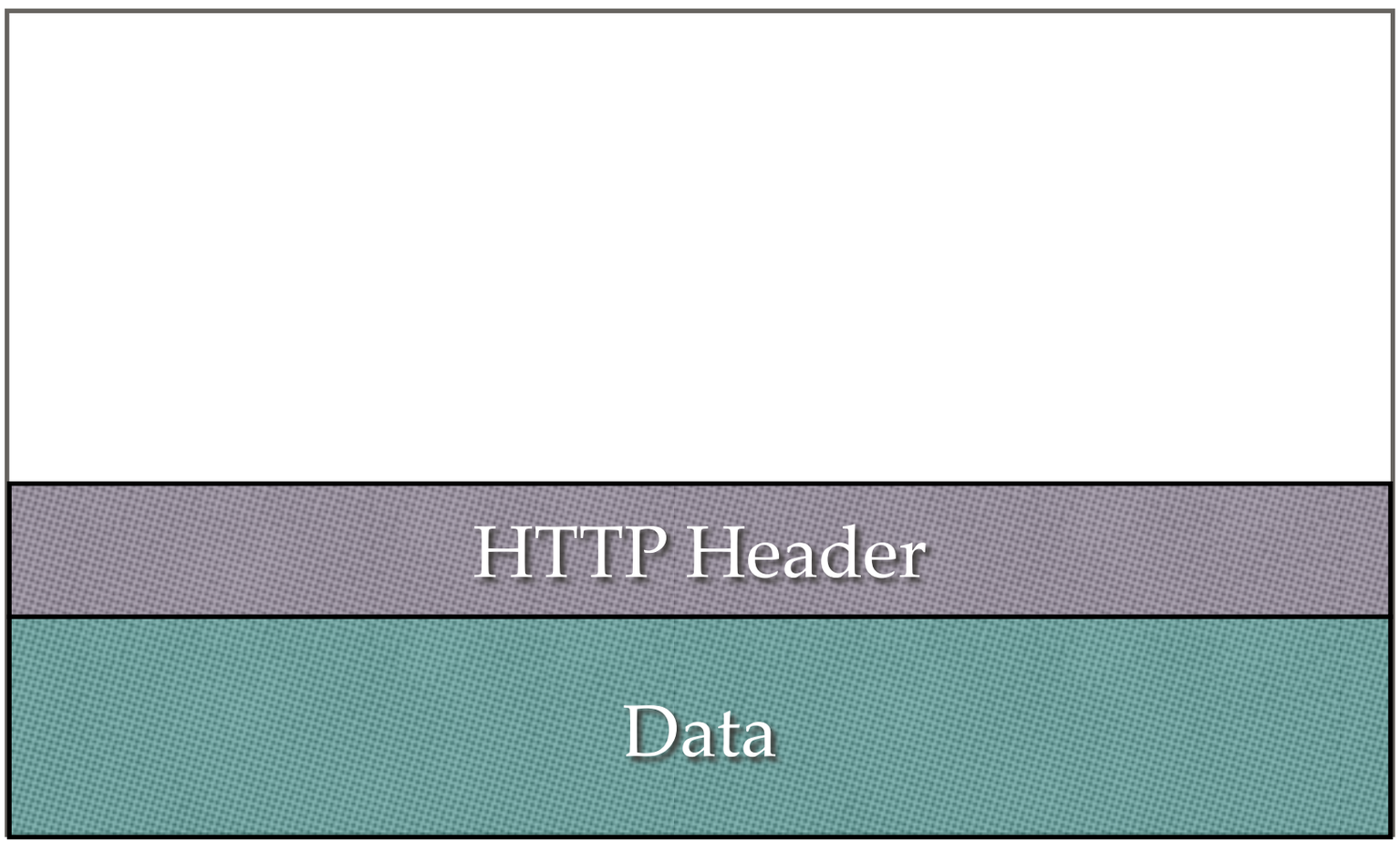
Packets

Data divided into packets to send across network

"LOLCats!"

Packets

Data divided into packets to send across network

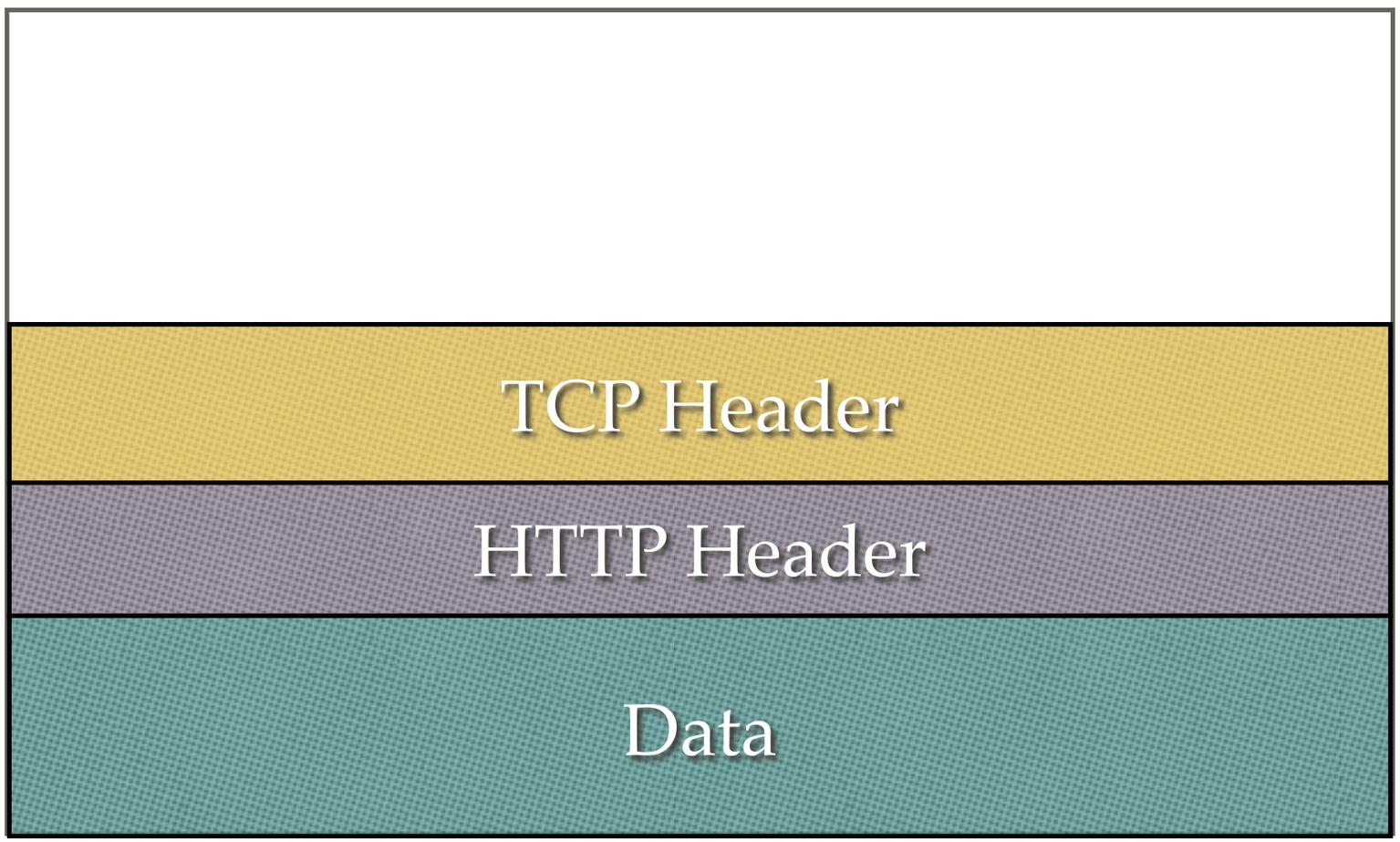


Response-header:

“LOLCats!”

Packets

Data divided into packets to send across network



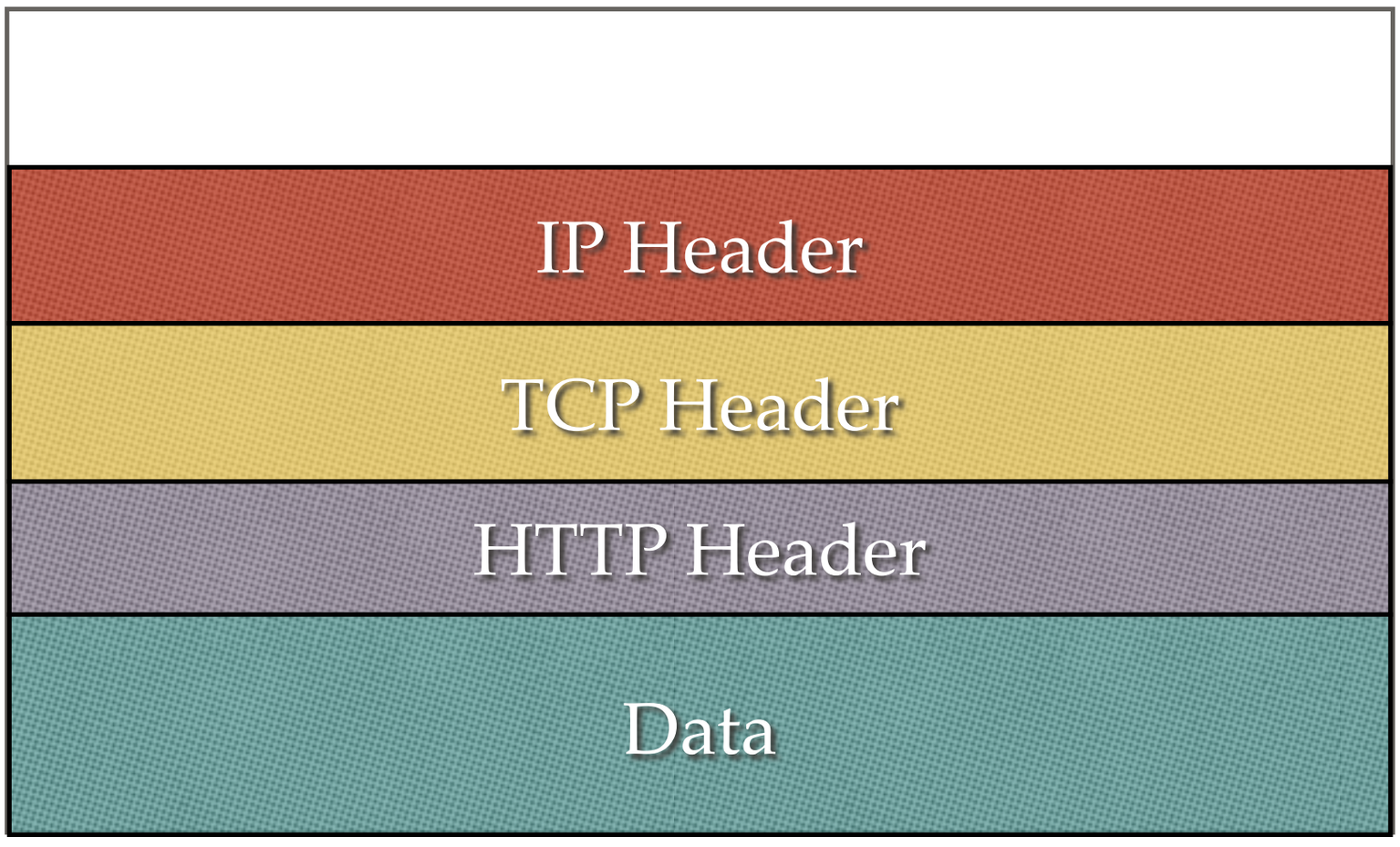
Port 23

Response-header:

“LOLCats!”

Packets

Data divided into packets to send across network



192.0.0.10

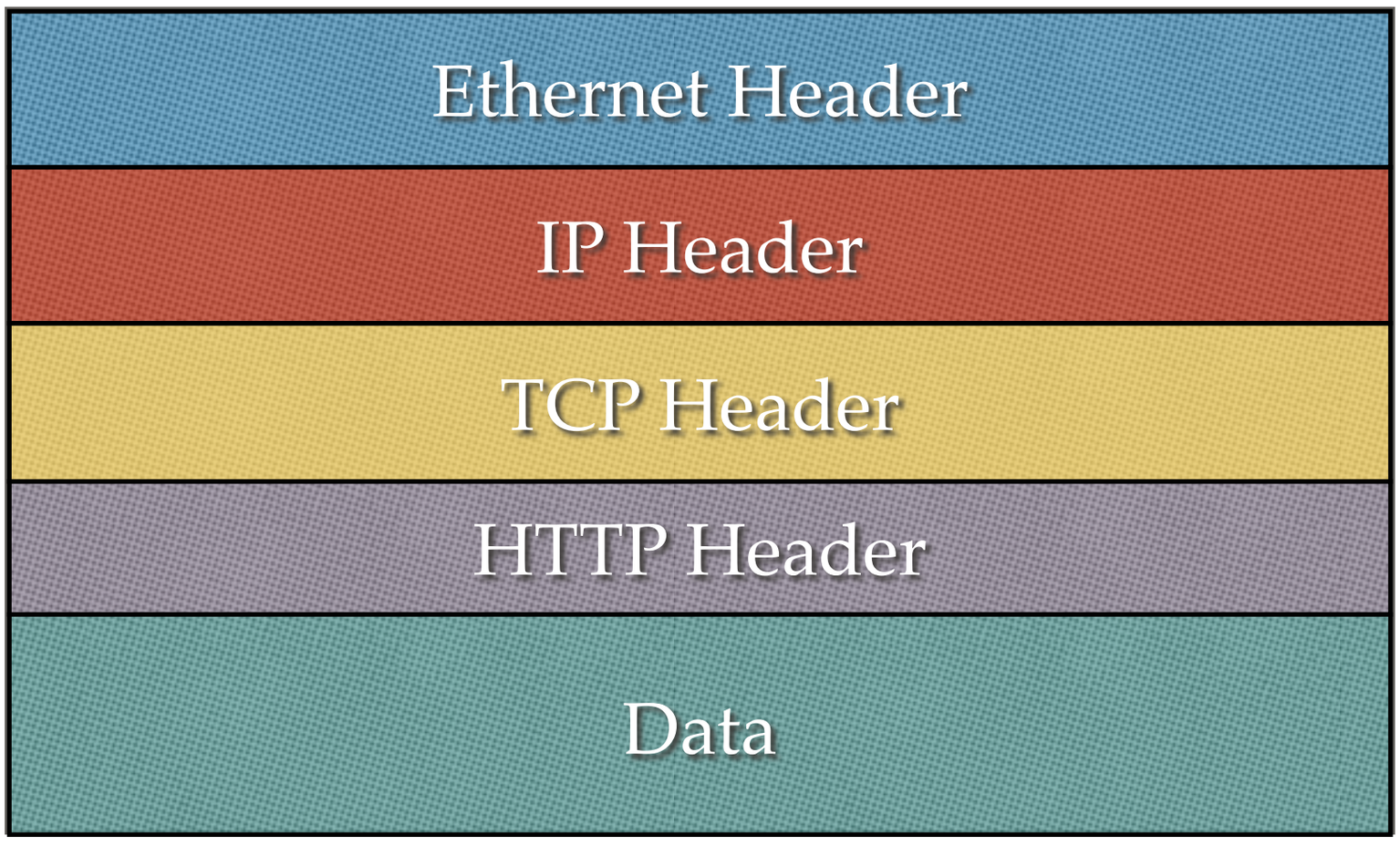
Port 23

Response-header:

“LOLCats!”

Packets

Data divided into packets to send across network



MAC 00:23:56:00:10:90

192.0.0.10

Port 23

Response-header:

“LOLCats!”

Types of Network Protocols

- Connection or No connections
  - Connections: Indicate one time who you are talking to
  - No connections: Each message specifies recipient
- Message boundaries vs. Streams
- Reliable vs. Unreliable
  - Reliable: Will retransmit lost data transparently
  - Unreliable: Sent data can be lost

Two APIs to Rule Them All

System V STREAMS

- Used in most commercial Unix variants (e.g., Solaris)
- More or less discontinued
- Implementation has same principles of sockets

Berkeley Sockets

- The standard interface for network applications
- Linux, FreeBSD, Mac OS X, AIX
- Emulated in Solaris and other System V Unices

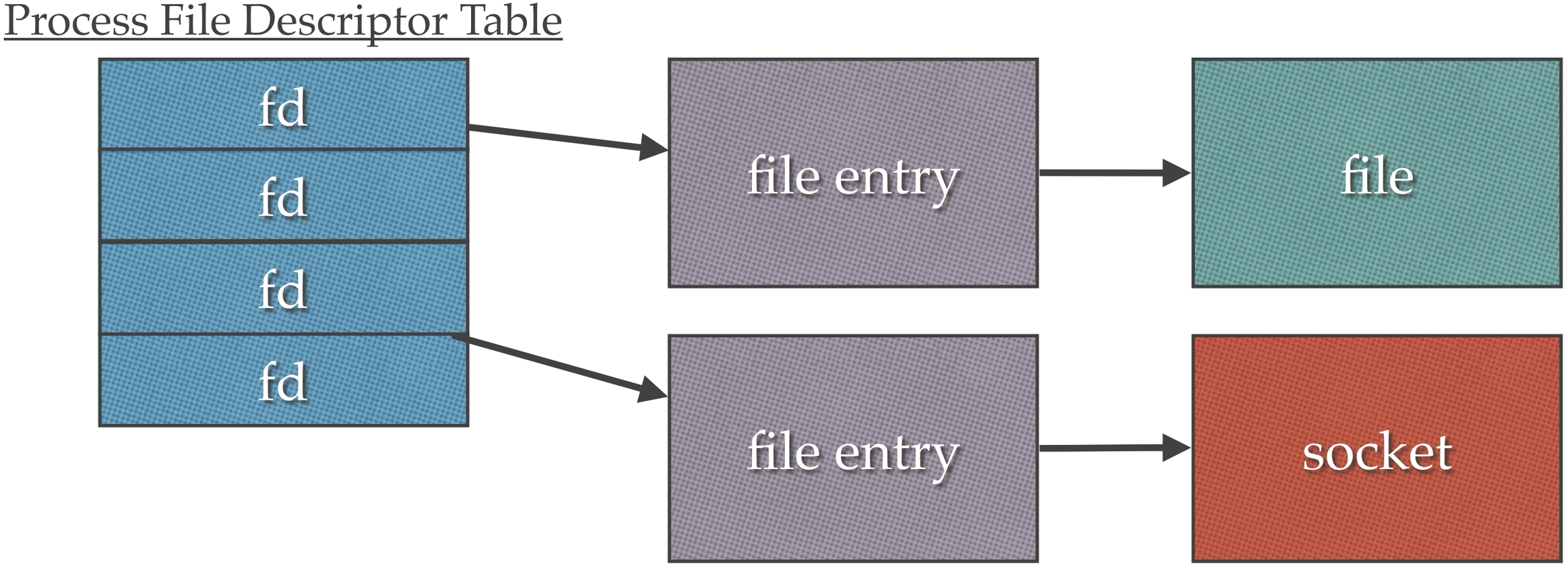
Berkeley Sockets Implementation

Socket: A Pipe for the Network

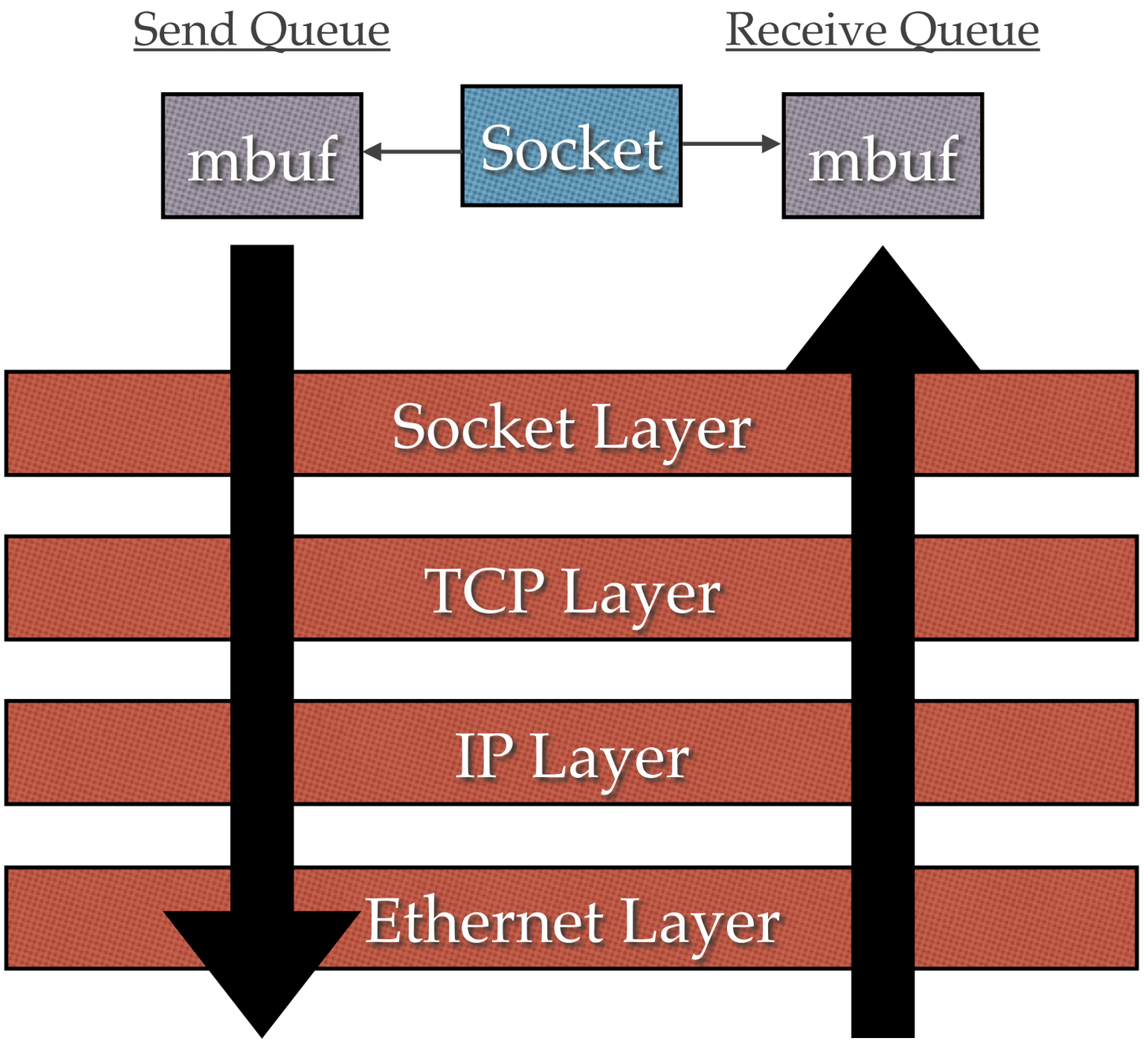
- Full-duplex: data moves each direction
- Created by socket() system call
- Kernel performs all protocol processing

Socket

- Socket is a file descriptor
- Can (usually) be used like a file



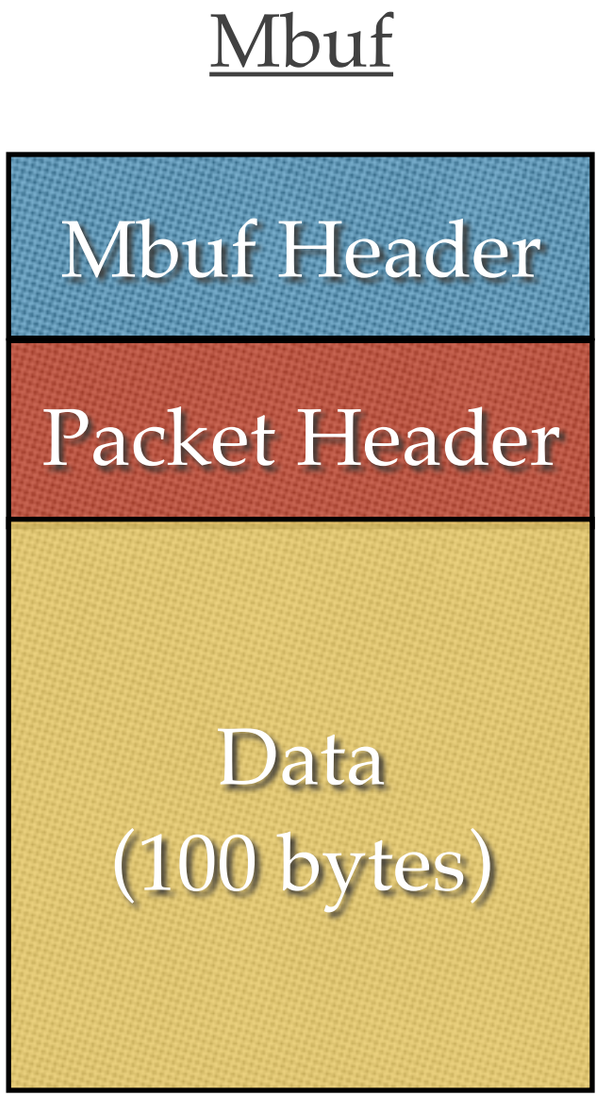
Down (and Up) the Rabbit Hole



The name of the game is to avoid copying data

The Almighty Message Buffer (mbuf)

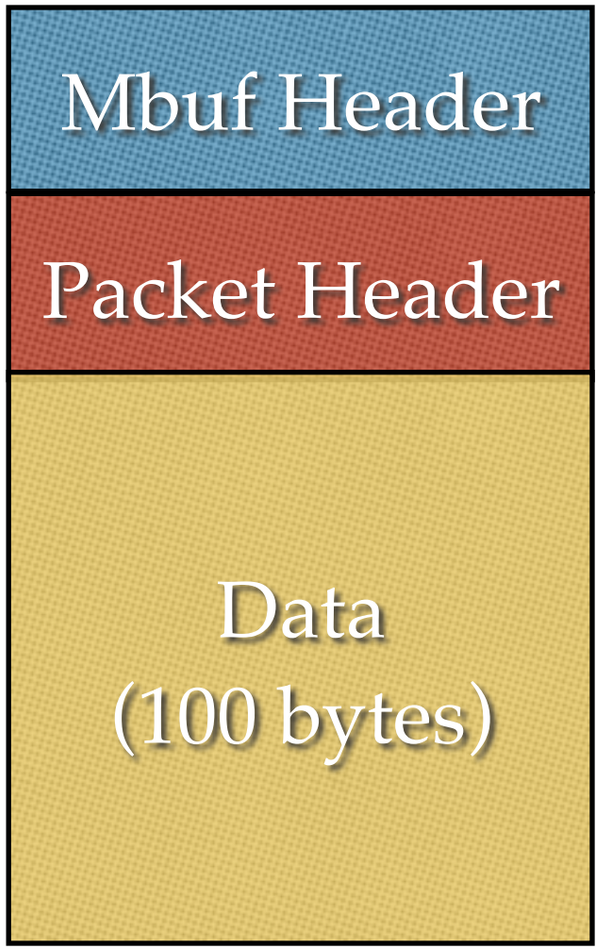
- Represents a network packet
- Mbuf Header
  - Pointer to start of data
  - Length of data
  - Pointer to next mbuf in chain
  - Pointer to next mbuf in queue



The Almighty Message Buffer (mbuf)

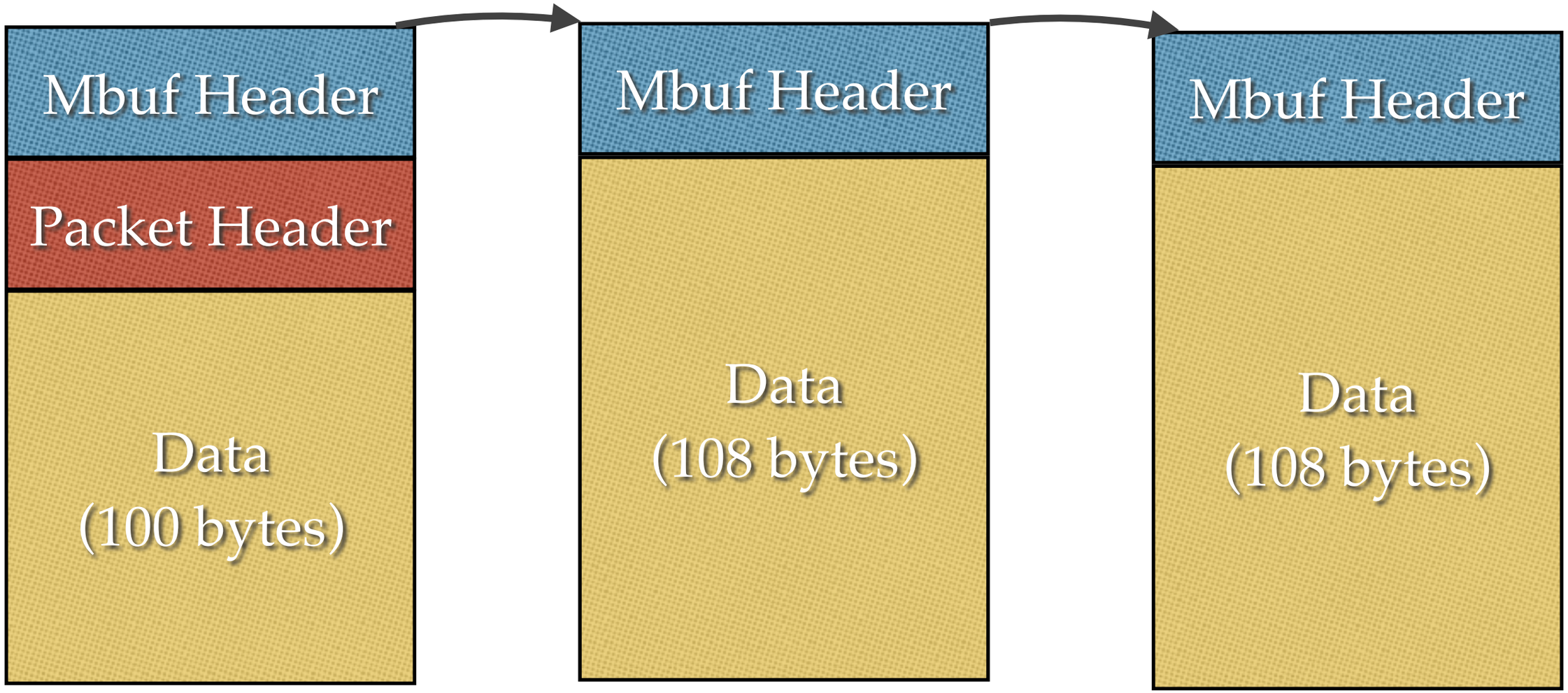
- Represents a network packet
- Packet Header Struct
  - *Not* IP Header
  - Only marks start of chain
- Data (payload) area

Mbuf



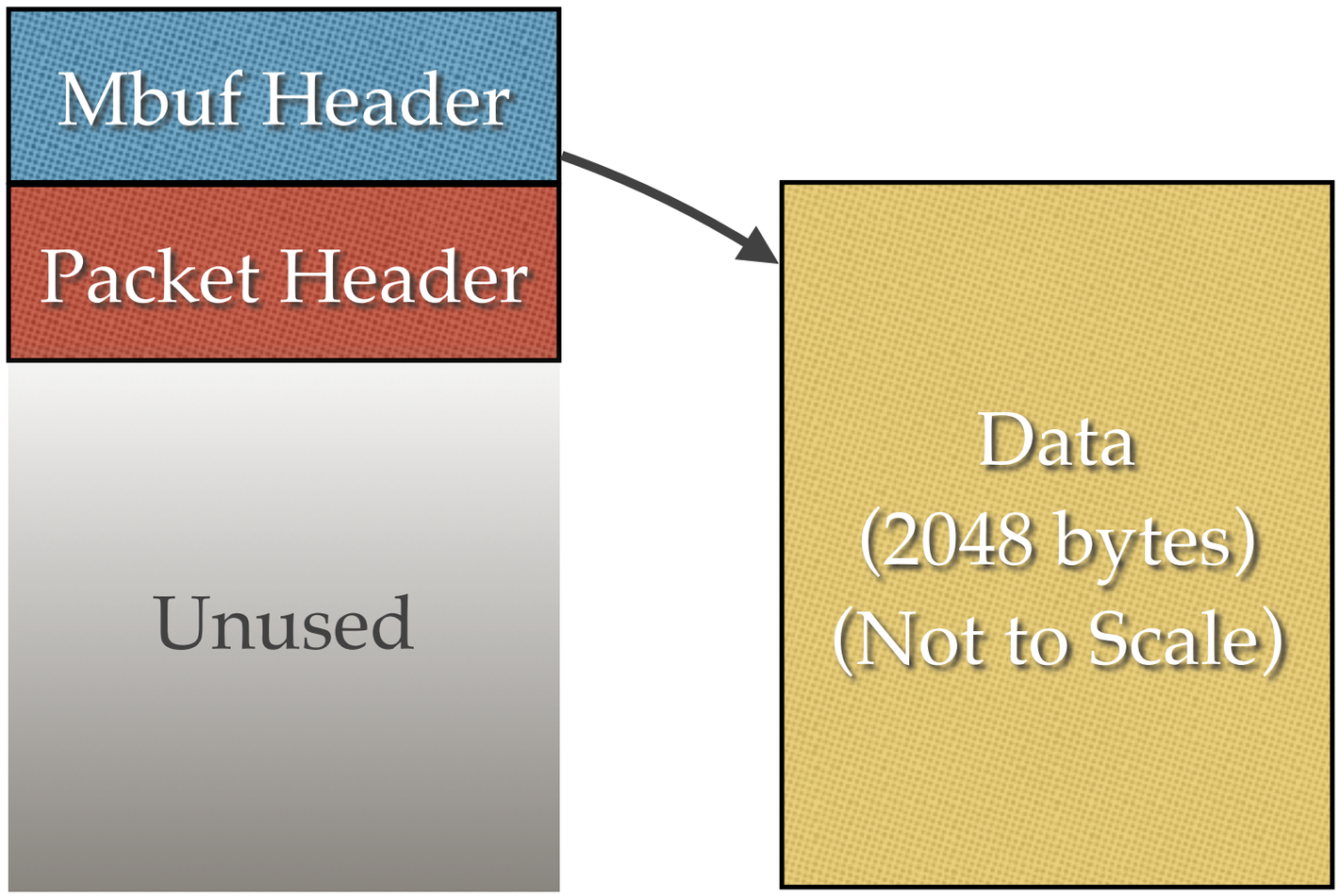
mbuf Chains

- mbufs can be chained together; chain represents ***one*** network packet
- First mbuf has packet header: marks beginning of chain



Cluster mbuf (mbuf on Steroids)

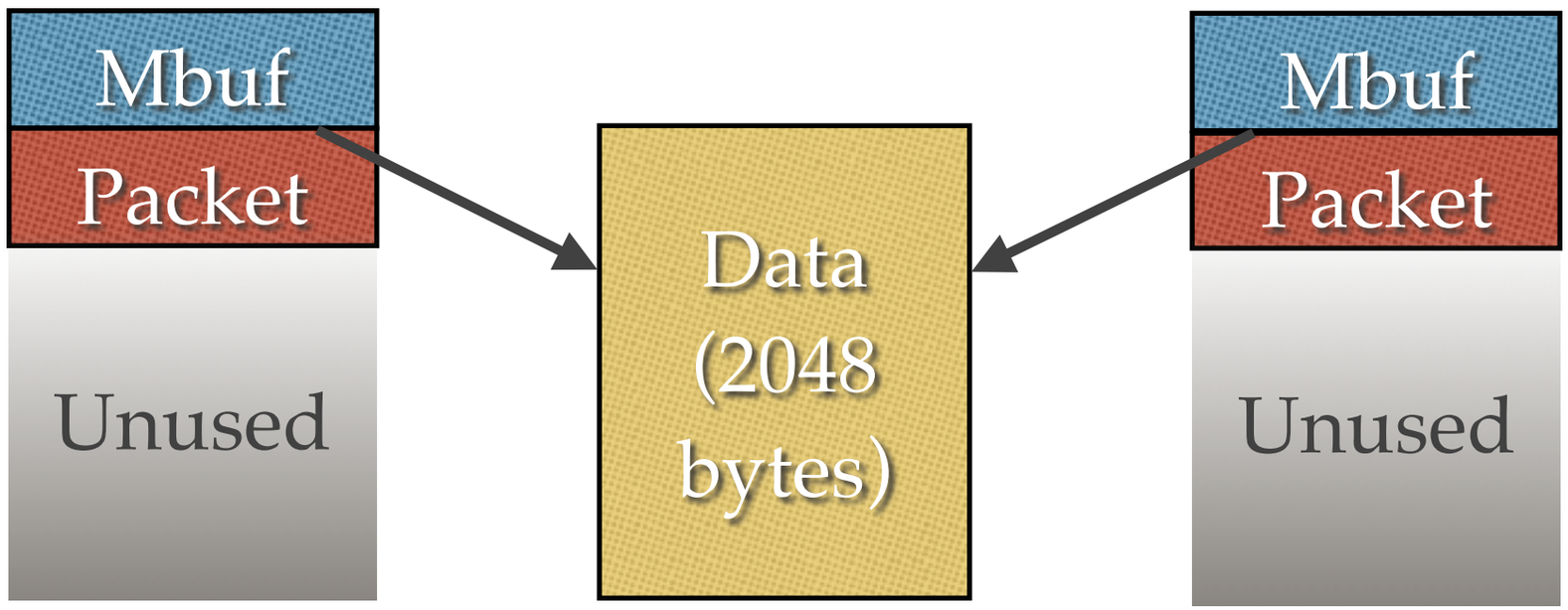
- Field in header points to externally allocated buffer
- Allows for mbuf with large data



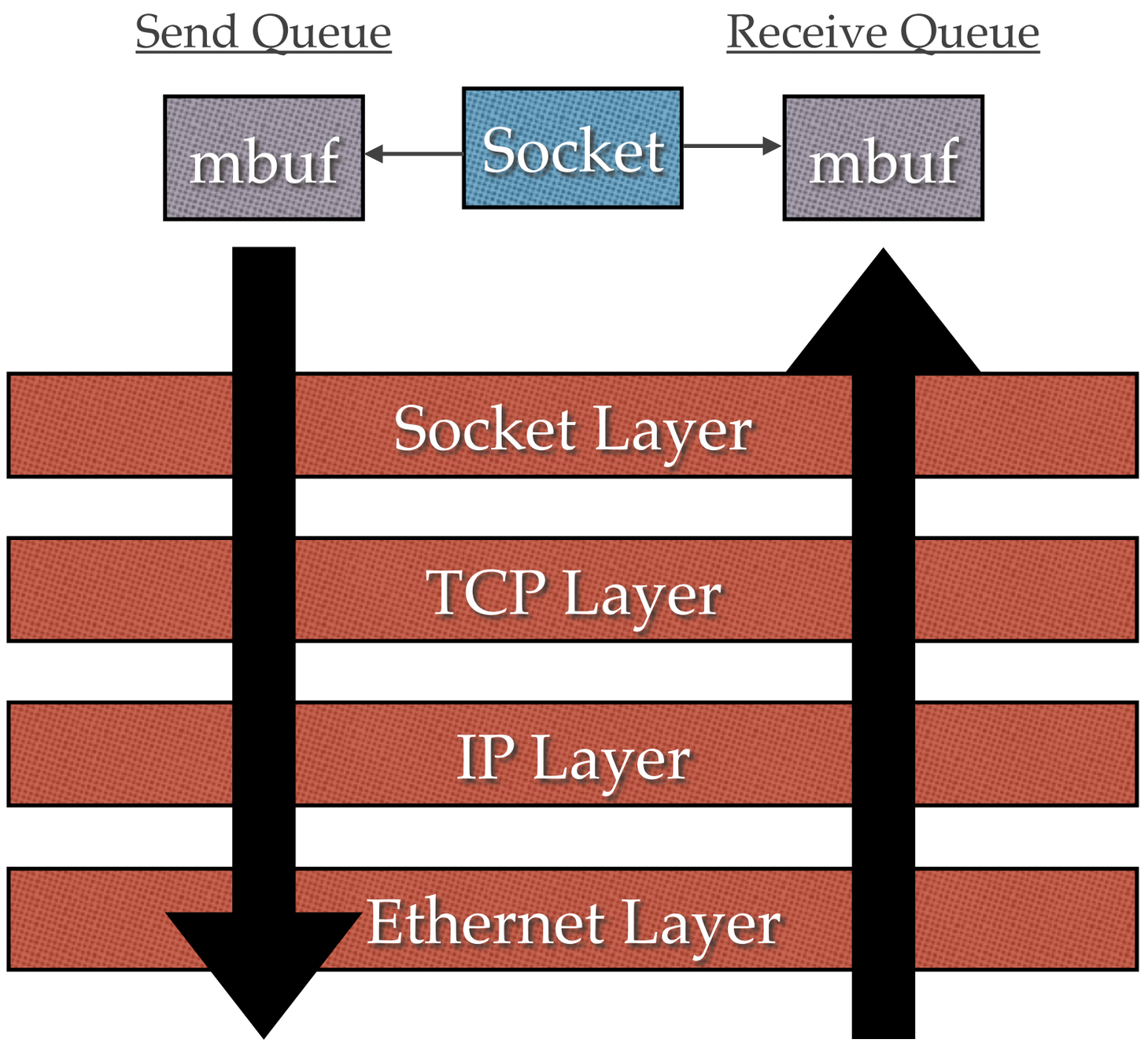
If you can chain mbufs, of what benefit is a cluster mbuf?

The Benefits of Clusters

- mbuf clusters can hold large amount of contiguous data
  - More efficient for large packets
  - More efficient for copying data between user and kernel space
- Clusters can be ***shared*** between mbufs

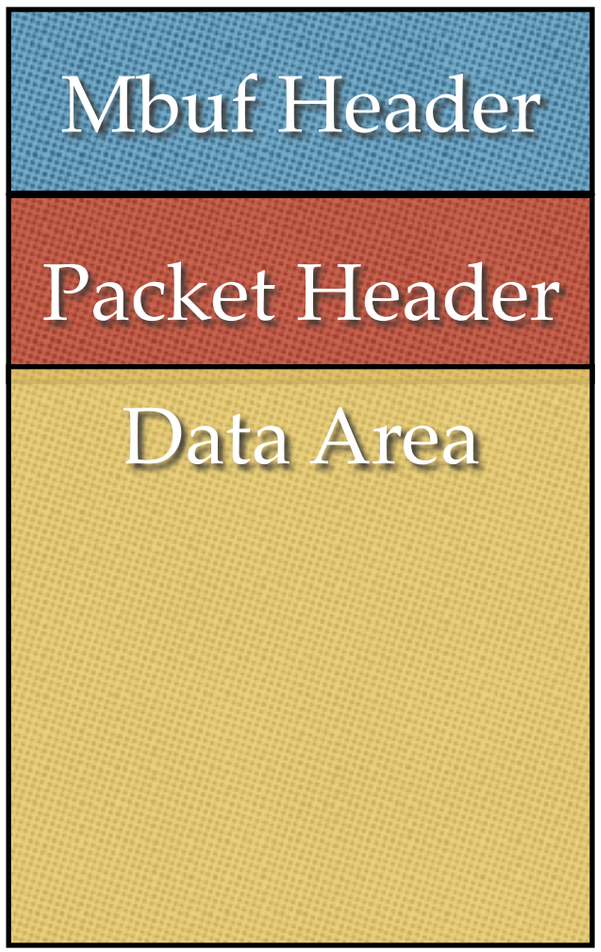


Down (and Up) the Rabbit Hole



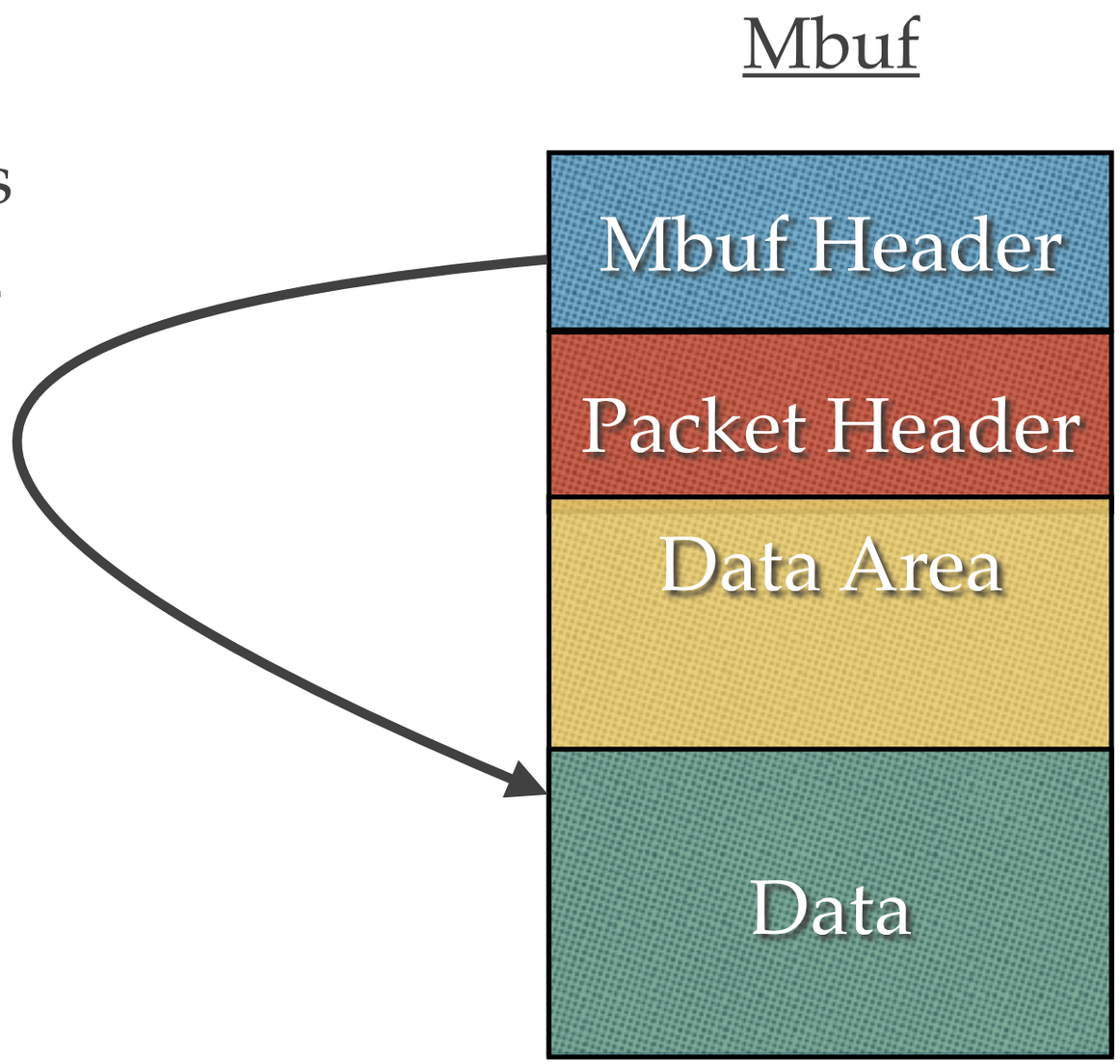
Sending Data

- Put data into mbuf
- Each layer prepends headers
  - Only adjusts data in mbuf header!



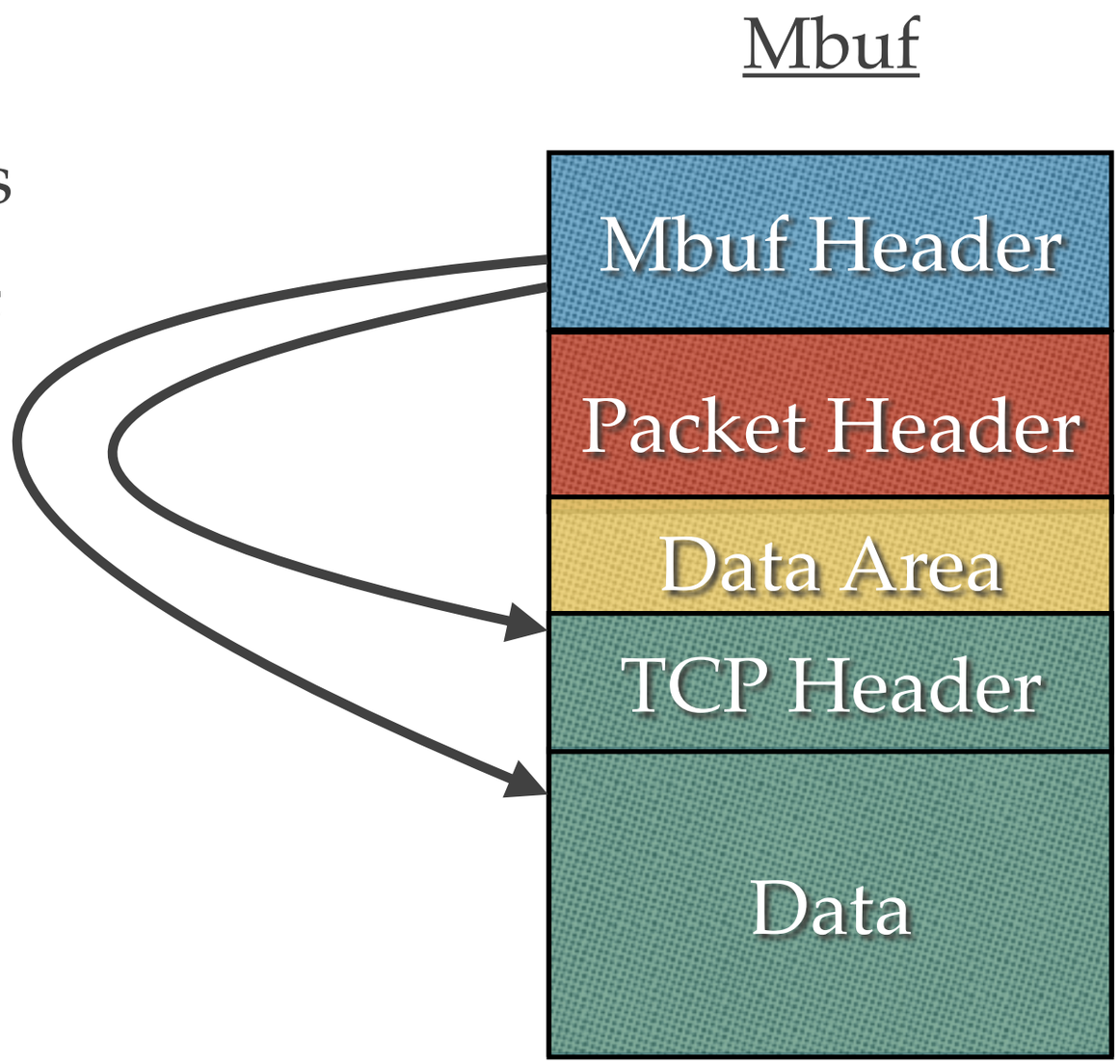
Sending Data

- Put data into mbuf
- Each layer prepends headers
  - Only adjusts data in mbuf header!



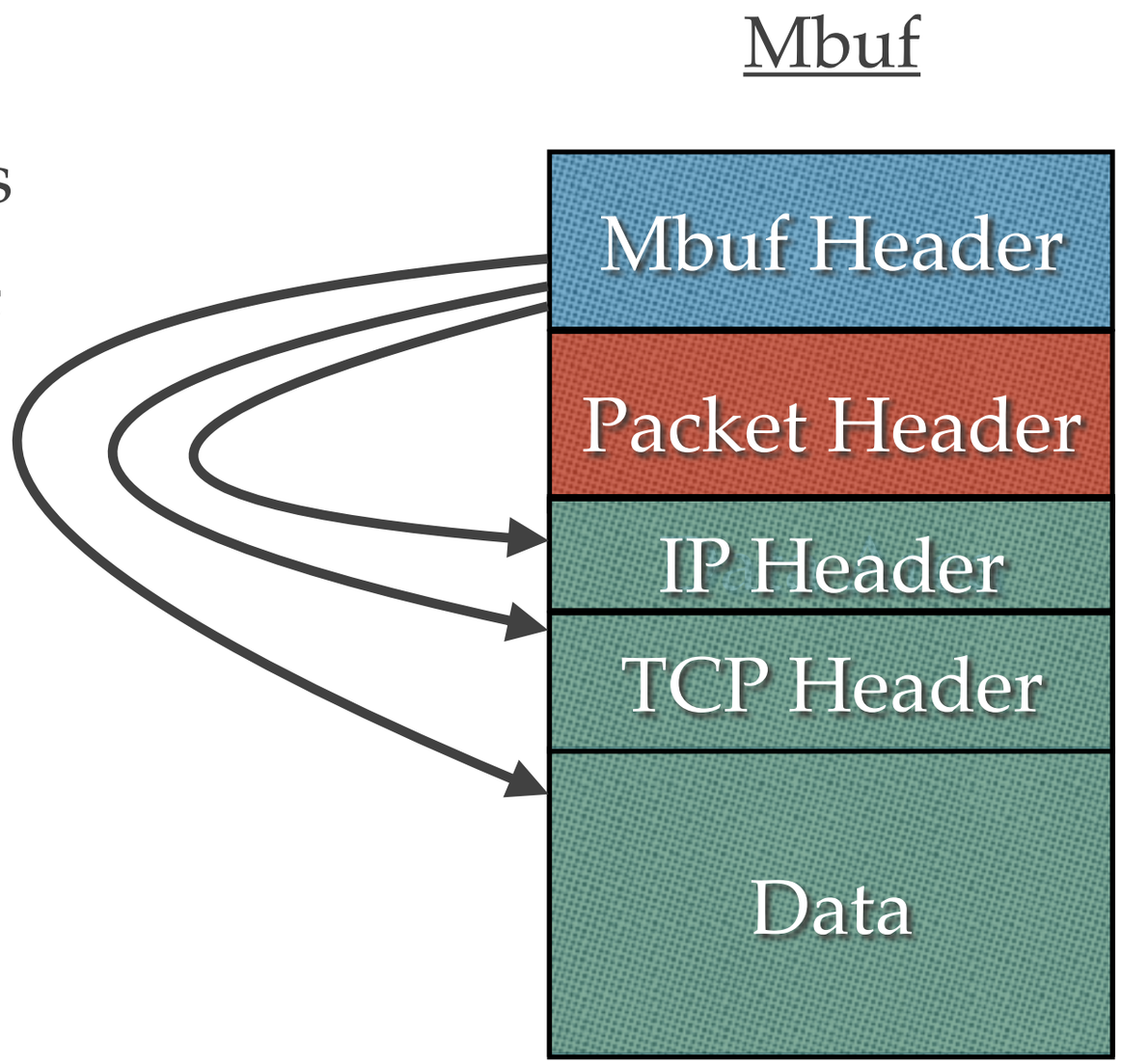
Sending Data

- Put data into mbuf
- Each layer prepends headers
  - Only adjusts data in mbuf header!



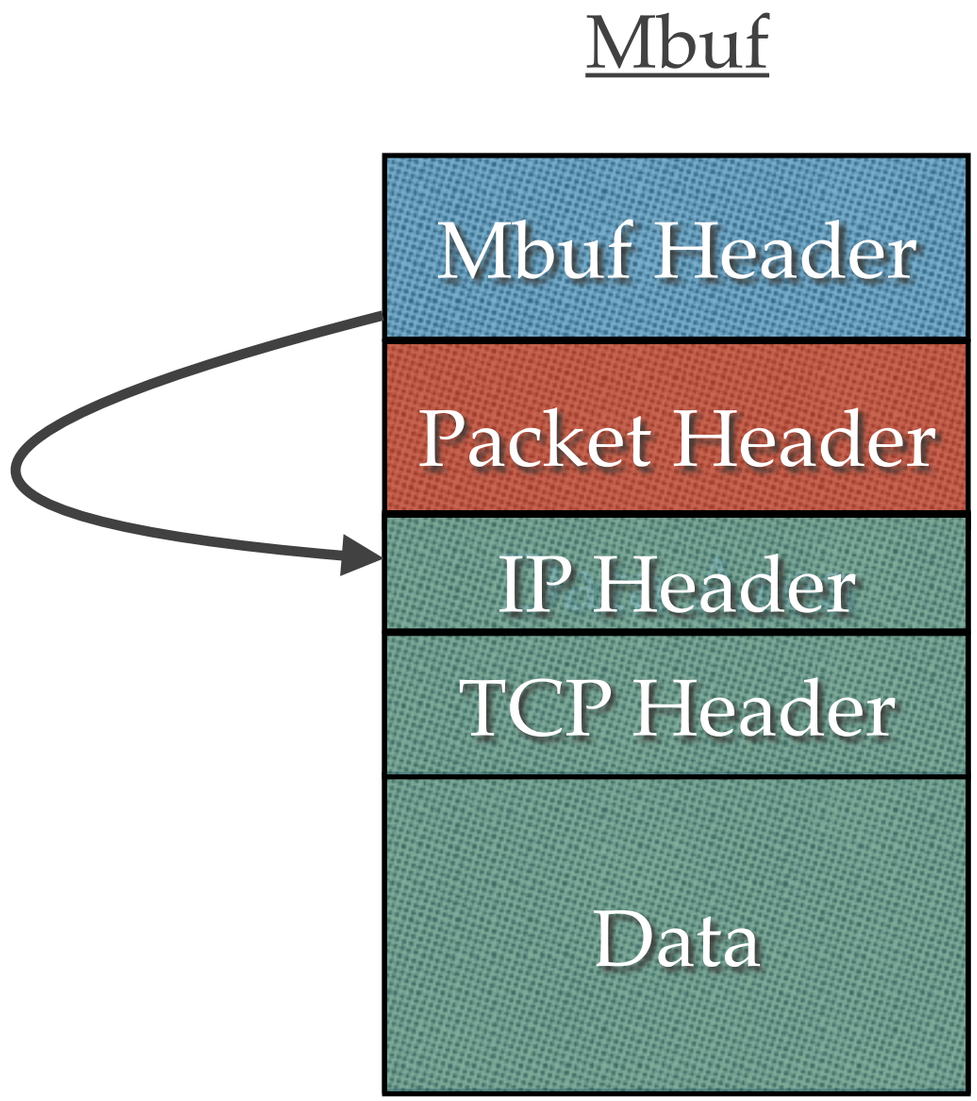
Sending Data

- Put data into mbuf
- Each layer prepends headers
  - Only adjusts data in mbuf header!



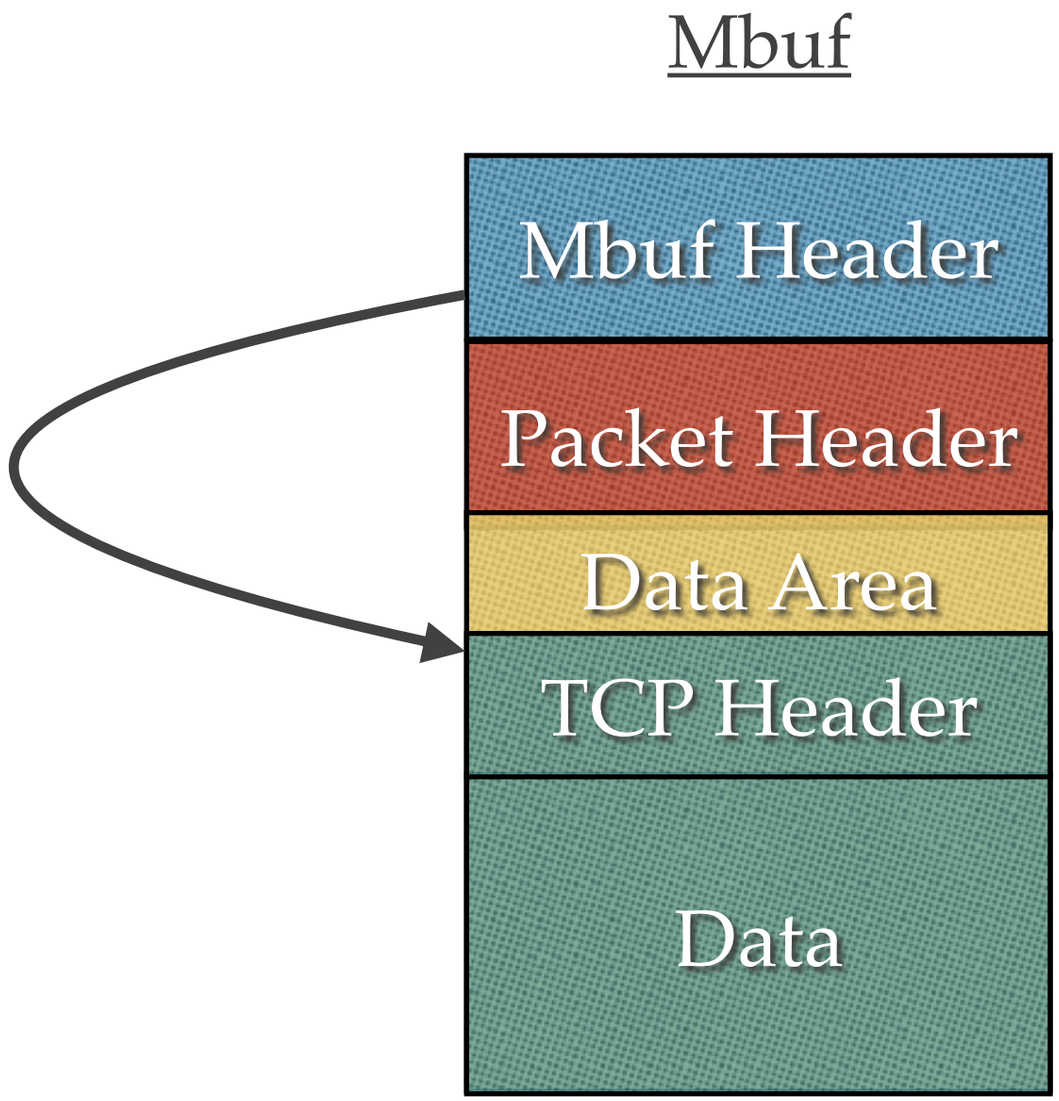
Receiving Data

- Remove headers from mbuf
- Only adjusts data in mbuf header!



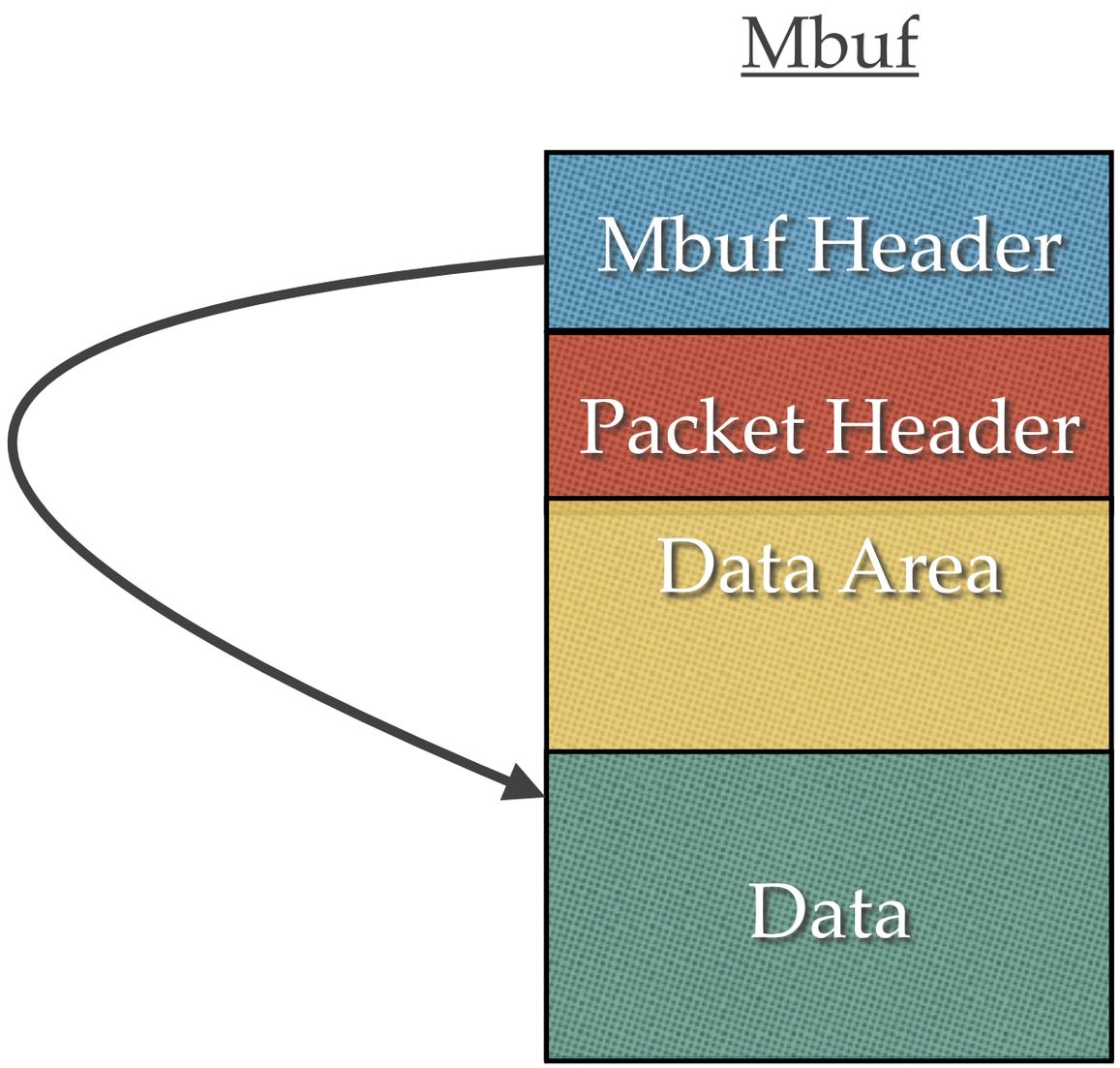
Receiving Data

- Remove headers from mbuf
- Only adjusts data in mbuf header!



Receiving Data

- Remove headers from mbuf
- Only adjusts data in mbuf header!



Berkeley Socket API

Creating Sockets

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

- Socket is a file descriptor
  - Created by socket system call
  - Full-duplex
- Domain determines protocol family (e.g, IP, AppleTalk)
- Type indicates whether messages have boundaries
- Protocol specifies a specific protocol within the family (TCP, UDP, ICMP, IGMP)

Setting the Endpoints

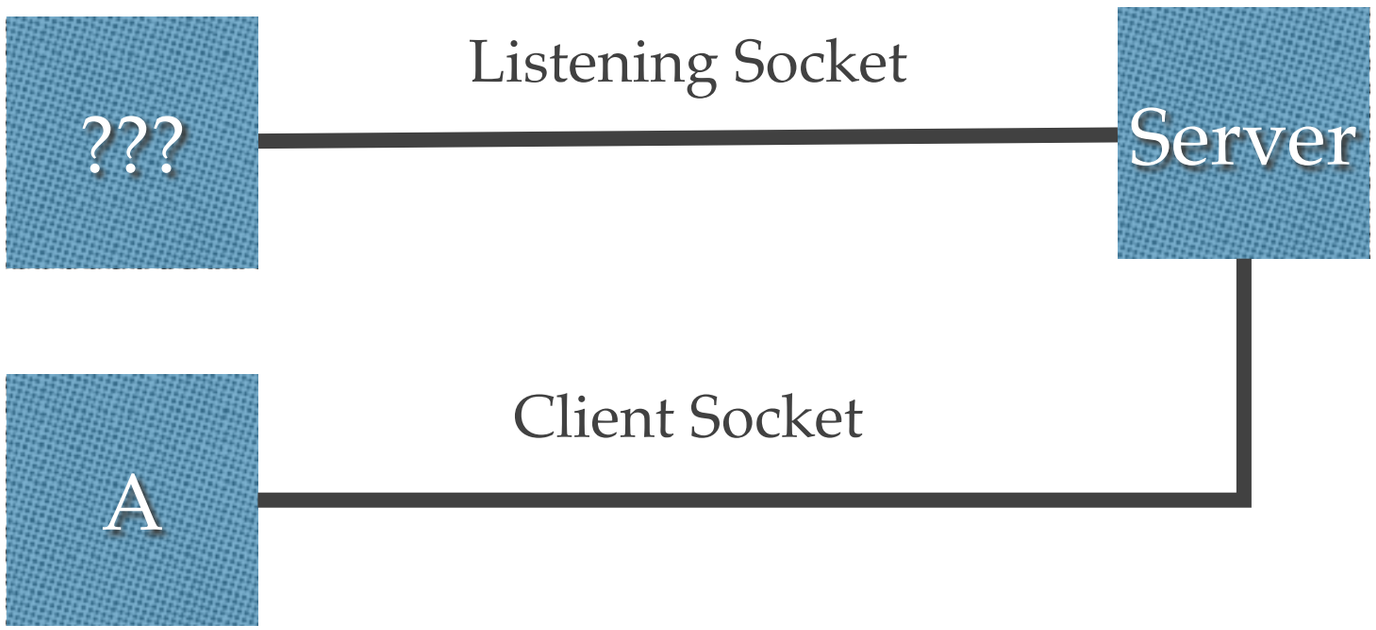
- bind(): Sets the address of the socket's local side
- connect(): Sets the address of the socket's remote side
  - For TCP, sends 3-way handshake

Establishing Connections

- Needed for connection-oriented protocols (e.g., TCP)
- connect(): Starts client 3-way handshake
- accept(): Accepts new connection for server
  - Returns new socket for that specific client
- Must be done for sending/receiving data

accept() in Pictures

accept() in Pictures



Berkeley Sockets: I/O

- Sockets are file descriptors
  - Can use read() and write()
  - Can use other system calls like dup(), close()
- Can use sendto() and recvfrom()
  - Specify to where data should go
  - Determine from whence data comes
  - Specify special options (non-blocking, out of band)

Typical TCP Client and Server

- Client
  - fd = socket (…)
  - connect (fd, …);
- Server
  - fd = socket (…);
  - bind (fd, …);
  - fd2 = accept (fd, …);

Server Architecture

Types of Servers

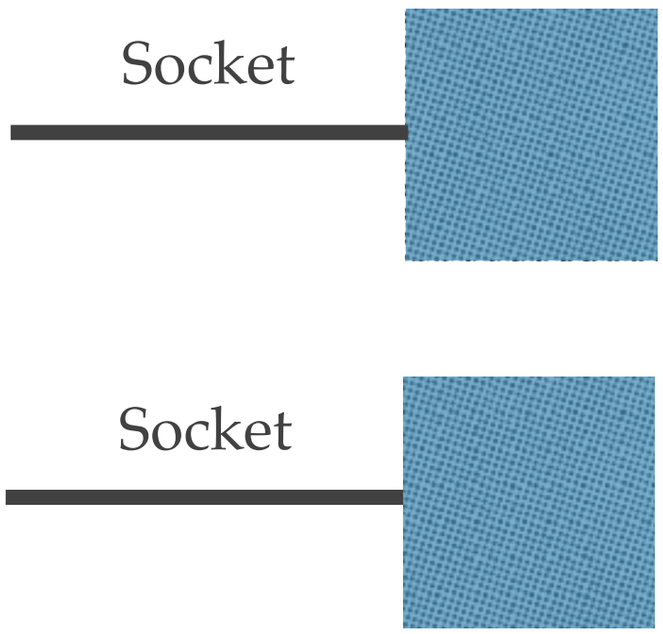
- telnetd
- ftpd
- dhcpd
- bootpd
- time
- echo
- daytime
- quotd
- chargen
- sshd
- httpd
- imapd
- popd
- sendmail (SMTP)
- nfsd
- bind (DNS)
- ircd
- ldapd

Fork on Accept

- Single process waits for a connection via accept()
- Create new process (thread) when a connection arrives

Fork on Accept

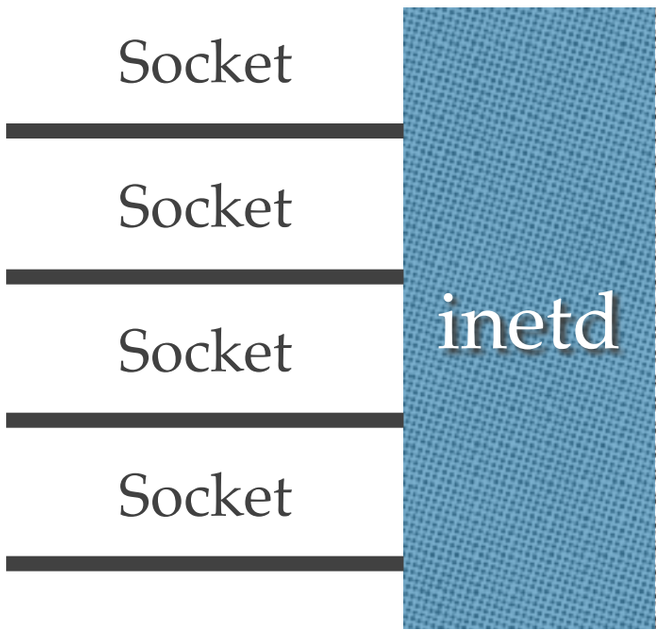
- Single process waits for a connection via accept()
- Create new process (thread) when a connection arrives



What happens if these servers are all separate programs?

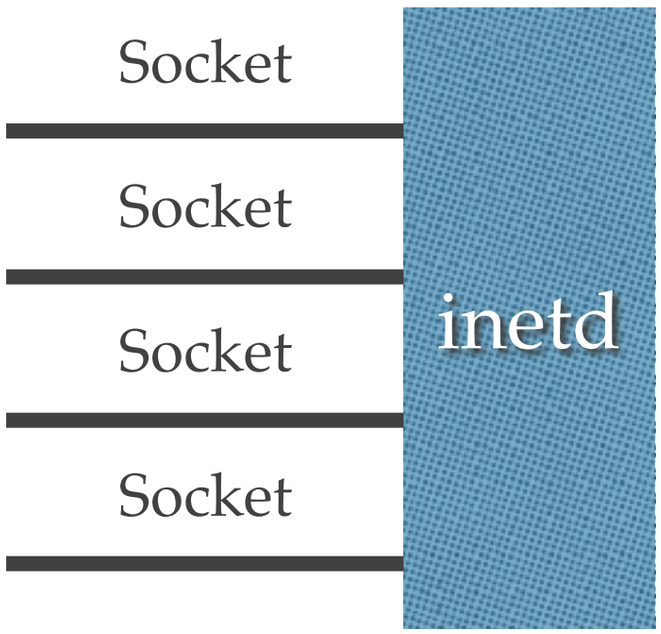
Internet Super Server (inetd)

- Use single process to wait for connections
- Hands socket off as stdin/stdout of programs
- Some requests handled by inetd directly



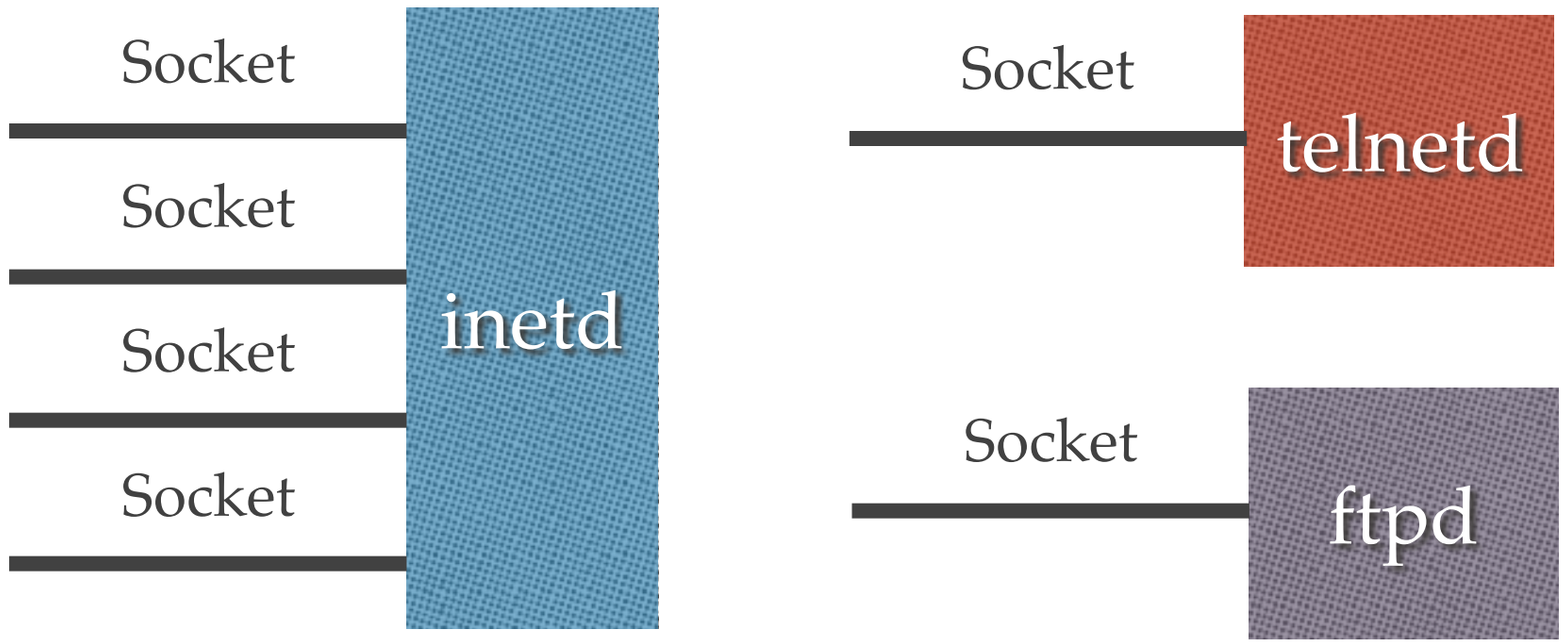
Internet Super Server (inetd)

- Use single process to wait for connections
- Hands socket off as stdin/stdout of programs
- Some requests handled by inetd directly



Internet Super Server (inetd)

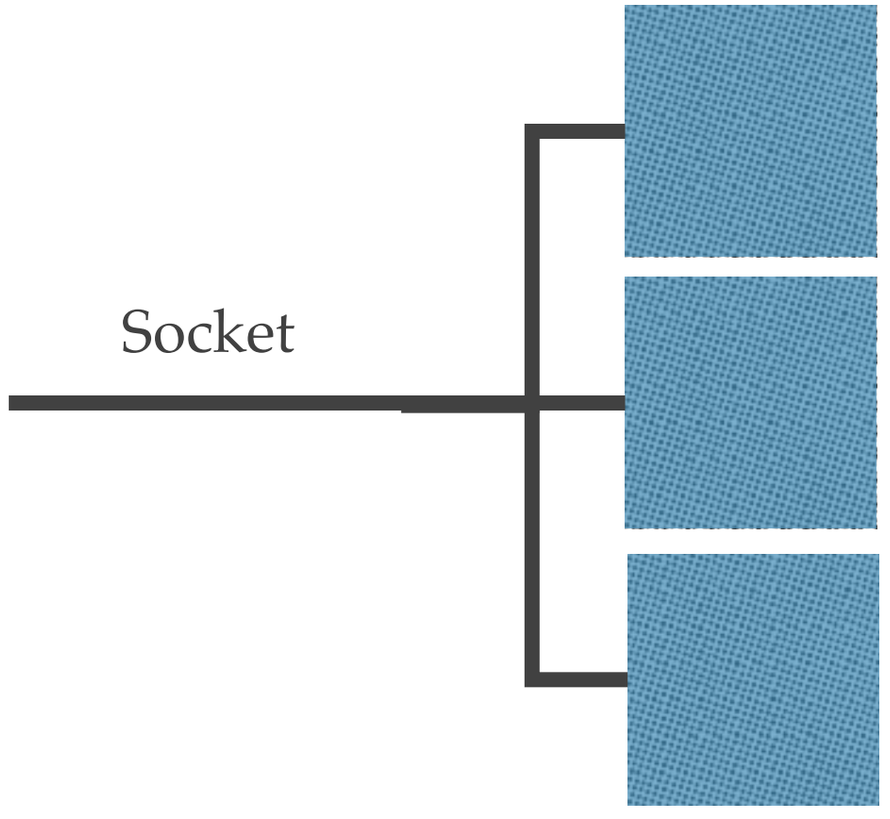
- Use single process to wait for connections
- Hands socket off as stdin/stdout of programs
- Some requests handled by inetd directly



What happens if you get *a lot* of connections (e.g., Google, the CSC 256/456 web site)?

Pre-forked/Pre-threaded

- Create multiple processes
- All processes wait on a single socket



What is the disadvantage of pre-forked/pre-threaded servers?

Security

- Pre-forked servers reuse processes
  - Stale data
  - Memory leaks
- Processes often replaced after awhile

Network Summary

- Networks are like onions (and ogres!): they have layers!
- Kernel must add/remove headers
  - Each layer solves a different problem
- Network data structures designed to minimize copies
  - mbuf designed specifically to achieve this
- Servers designed for performance and security