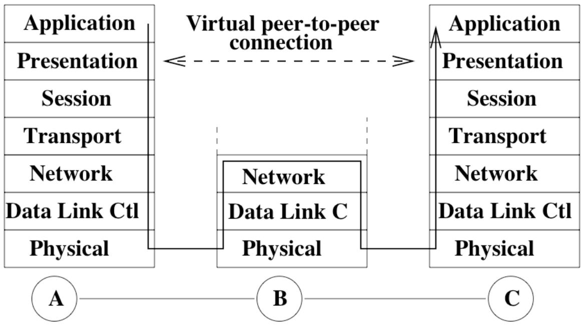
Unix and Minix Networking

- Network Protocols
- Unix networking
- Minix networking

Communication Protocols

- Protocol: a set of rules for communication that are agreed to by all parties
- Protocol stack : networking software is structured into layers
  - Each layer N, provides a service to layer N+1, by using its own layer N procedures and the interface to the N-1 layer.
  - Example: International Standards Organization/ Open Systems Interconnect (ISO/OSI)



Application
Presentation
Session
Transport
Network
Data Link Ctl
Physical

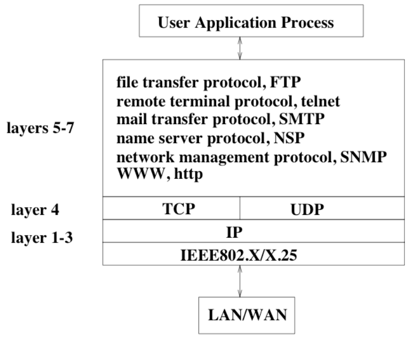
Virtual peer-to-peer connection

Network
Data Link C
Physical

Application
Presentation
Session
Transport
Network
Data Link Ctl
Physical

A
B
C

TCP/IP Protocol Stack



User Application Process

layers 5-7

file transfer protocol, FTP
remote terminal protocol, telnet
mail transfer protocol, SMTP
name server protocol, NSP
network management protocol, SNMP
WWW, http

layer 4 | TCP | UDP

layer 1-3 | IP

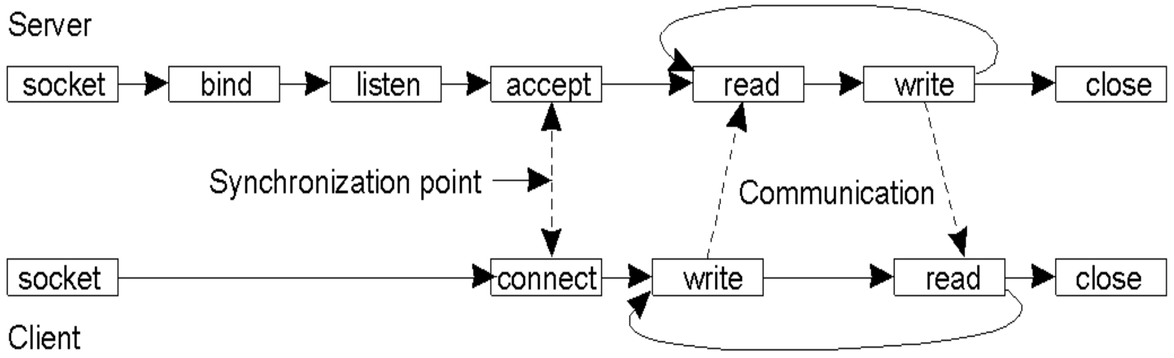
IEEE802.X/X.25

LAN/WAN

- Most Internet sites use TCP/IP - Transmission Control Protocol/ Internet Protocol.
  - It has fewer layers than ISO to increase efficiency.
  - Consists of a suite of protocols: UDP, TCP, IP...
  - TCP is a **reliable** protocol -- packets are received in the order they are sent
  - UDP (user datagram protocol) an **unreliable** protocol (no guarantee of delivery).

Socket Communication

- Client-server socket communication and Socket primitives
  - Berkeley sockets (BSD Unix)



Server

socket → bind → listen → accept → read → write → close

Synchronization point

Communication

socket → connect → write → read → close

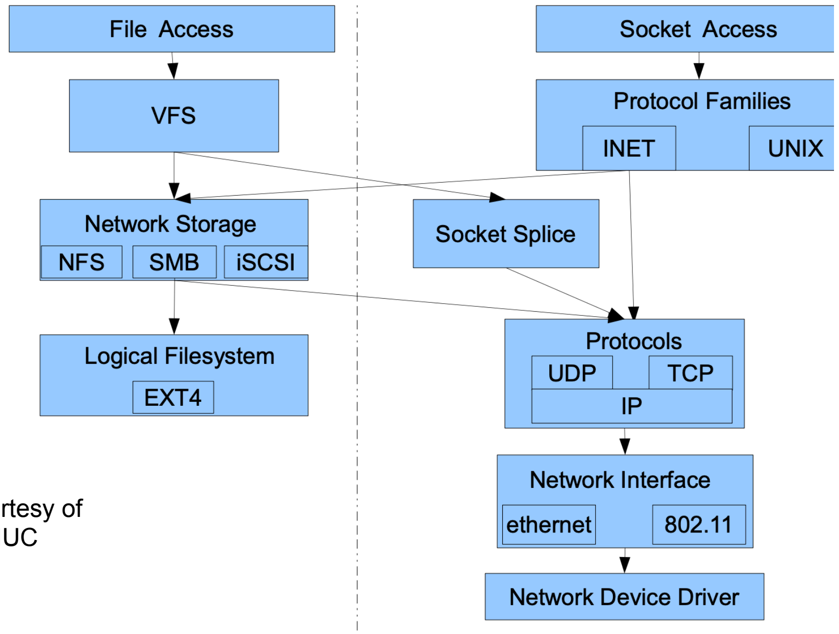
Client

Berkeley Socket Primitives

Primitive	Meaning
Socket	Create a new communication endpoint
Bind	Attach a local address to a socket
Listen	Announce willingness to accept connections
Accept	Block caller until a connection request arrives
Connect	Actively attempt to establish a connection
Send	Send some data over the connection
Receive	Receive some data over the connection
Close	Release the connection

Linux Network Architecture

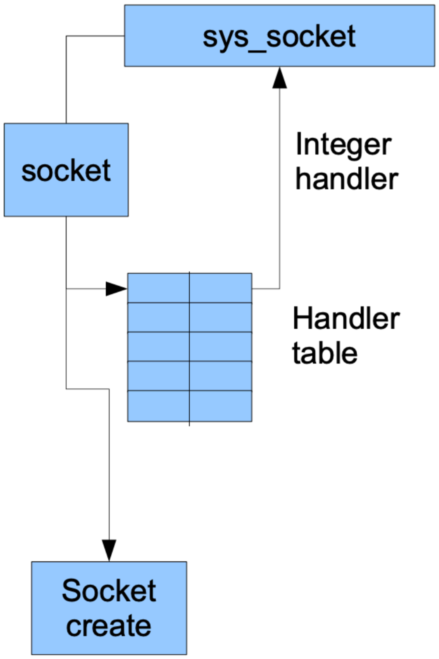
- File access path versus socket access path



Slides 6-12 courtesy of
Raoul Rivas, UIUC

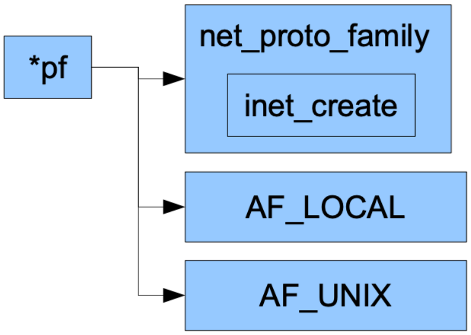
Sockets in Linux kernel

- Contains sys calls like socket, connect, accept
- Implements POSIX socket interface
  - independent of protocols
- Maps socket data structures to integer handlers
- Calls lower layer functions
  - sys_socket()->sock_create



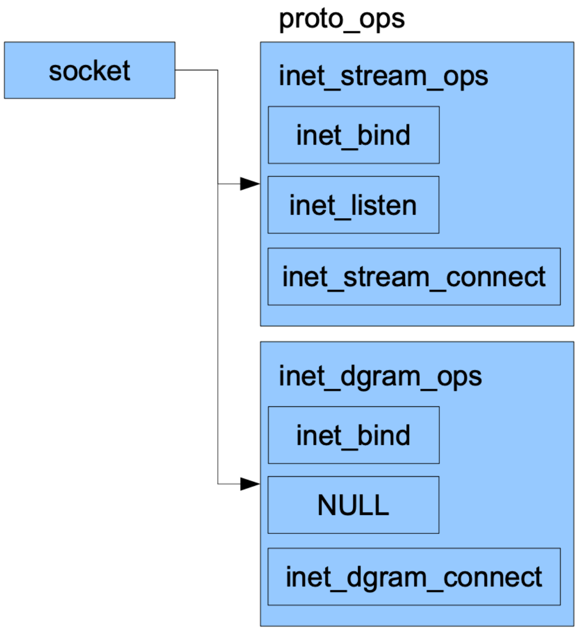
Protocol Families

- Implements different socket families: INET, UNIX
- Extensible through modules and fn pointers
- Calls net_proto_family->create for family-specific initialization



Protocols

- Families have multiple protocols
  - INET: TCP, UDP
- Protocol functions stores in proto_ops
- Some functions unused in a protocols: dummy fns
- Some functions same across protocols: shared



socket

proto_ops

inet_stream_ops

inet_bind

inet_listen

inet_stream_connect

inet_dgram_ops

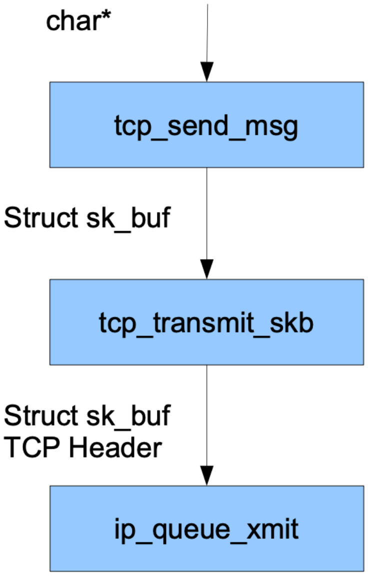
inet_bind

NULL

inet_dgram_connect

Packet Creation

- At sending function, packetize the buffer
- Packets represented as sk_buff data structure
- Contains pointers to
  - transport layer header
  - link layer header
  - received timestamp
  - Device that received it



char*

tcp_send_msg

Struct sk_buf

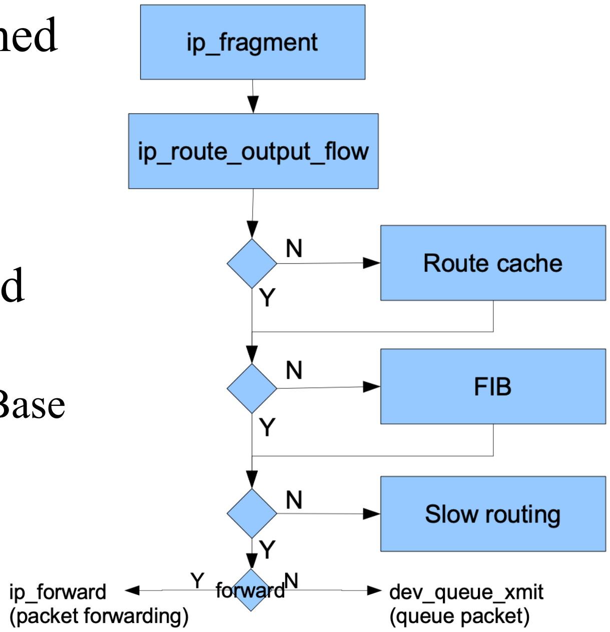
tcp_transmit_skb

Struct sk_buf
TCP Header

ip_queue_xmit

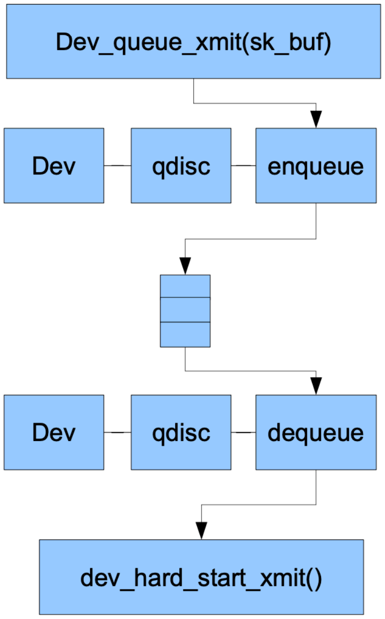
Fragmentation and Routing

- Fragmentation is performed inside ip_fragment
- route filled in by ip_route_output_flow
- Routing mechanisms used
  - Route cache
  - Forwarding Information Base
  - Slow routing



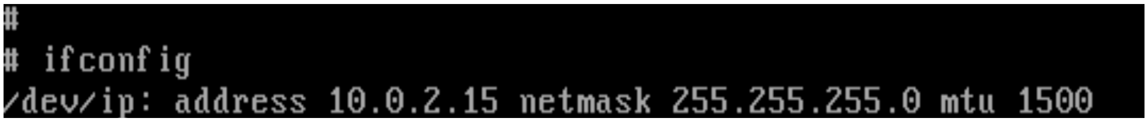
Data Link Layer

- Responsible for packet scheduling
- dev_queue_xmit enqueues packets for transmission
  - qdisc of device
- Send in process context
- If device bust, schedule for late
- dev_hard_start_xmit sends to device



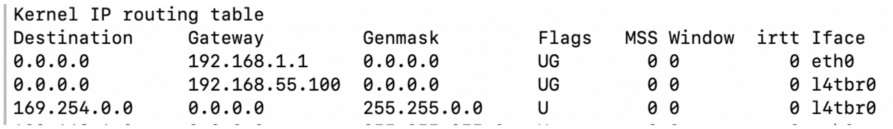
*NIX Networking Commands

- Ethernet MAC address: d0:73:d5:2a:12:51
- IP address: 192.168.1.2 or 128.119.240.2
- ping
- ifconfig



```
#
# ifconfig
/dev/ip: address 10.0.2.15 netmask 255.255.255.0 mtu 1500
```

- Linux: netstat -rn
- Linux: route



```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG        0 0          0 eth0
0.0.0.0         192.168.55.100  0.0.0.0         UG        0 0          0 l4tbr0
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 l4tbr0
```

Minix INET

- “inet” system process handles networking in Minix
  - Source code “servers/inet”
- Implements ethernet layer, IP layer and TCP/UDP

- Ethernet card is a I/O device
  - Device driver is in “drivers”
  - e1000 is Intel gigabit driver

- TCP/IP code is in “inet” and “inet/generic”

- http://www.nyx.net/~ctwong/minix/ note: minix v2, not v3

Data link Layer

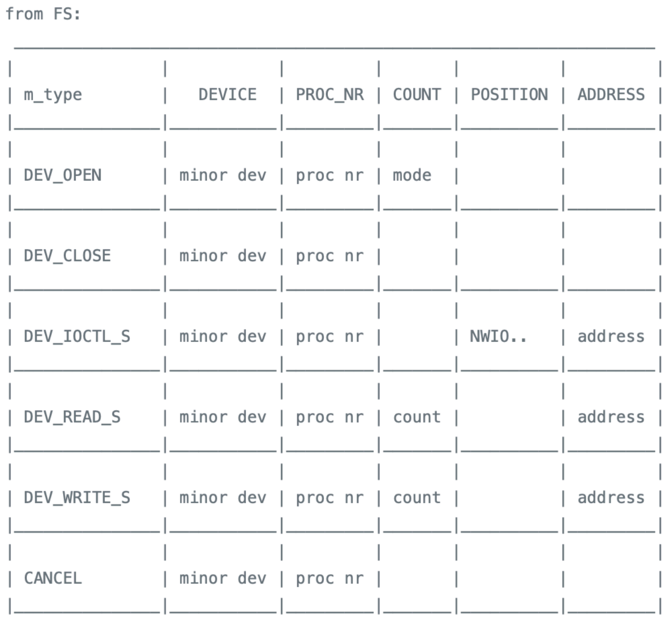
- Hardware: ethernet, modem etc
- Can have more than one device (major and minor #)
- ioctl call used to set parameters such as comm speed

- The driver itself runs as a user process

- I/O Involves: VFS, INET and driver process
  - same concept as any block device driver

INET Server

- inet.c - main function for INET Server
  - handles various message types from VFS and DL_ETH



```
from FS:

 ______________________________________________________________________
|               |           |          |       |          |         |
| m_type        |   DEVICE  | PROC_NR  | COUNT | POSITION | ADDRESS |
|_______________|___________|__________|_______|__________|_________|
|               |           |          |       |          |         |
| DEV_OPEN      | minor dev | proc nr  | mode  |          |         |
|_______________|___________|__________|_______|__________|_________|
|               |           |          |       |          |         |
| DEV_CLOSE     | minor dev | proc nr  |       |          |         |
|_______________|___________|__________|_______|__________|_________|
|               |           |          |       |          |         |
| DEV_IOCTL_S   | minor dev | proc nr  |       | NWIO..   | address |
|_______________|___________|__________|_______|__________|_________|
|               |           |          |       |          |         |
| DEV_READ_S    | minor dev | proc nr  | count |          | address |
|_______________|___________|__________|_______|__________|_________|
|               |           |          |       |          |         |
| DEV_WRITE_S   | minor dev | proc nr  | count |          | address |
|_______________|___________|__________|_______|__________|_________|
|               |           |          |       |          |         |
| CANCEL        | minor dev | proc nr  |       |          |         |
|_______________|___________|__________|_______|__________|_________|
```

INET Server

- buf.c - buffering code to allocate data for sending and receiving network packets
- mnx_eth.c - code for sending and receiving ethernet frames to/from ethernet driver
- inet_config.c - configure networking devices
  - /dev/eth, /dev/ip, dev/tcp, /dev/udp
- mq.c — message queue structure
  - mq_list is message queue and mq_t is one message entry

INET Server

- sr.c - code to interface with file system
  - DEV_OPEN, DEV_CLOSE, DEV_READ, DEV_WRITE…
- generic/udp.c - code for UDP protocol
  - udp_port data structure is used for a UDP socket port
- generic/tcp.c - code for TCP protocol
  - tcp_port data structure used to TCP socket port