CSCI 6907.11

Adv. Net. Sys. Prog.

Lecture 4 - Network + OS

Tim Wood
CS@GWU
2015

Some content from Kurose and Ross

Today

Homework Review

- Swap tables and share code

Protocols

- IP, TCP, UDP

Measuring network performance

- latency, throughput, drop rate

Tracing programs

- ltrace, strace on hello world and sendtcp

The path of a packet

- HW, OS, software

Git merging and branching

Homework 1

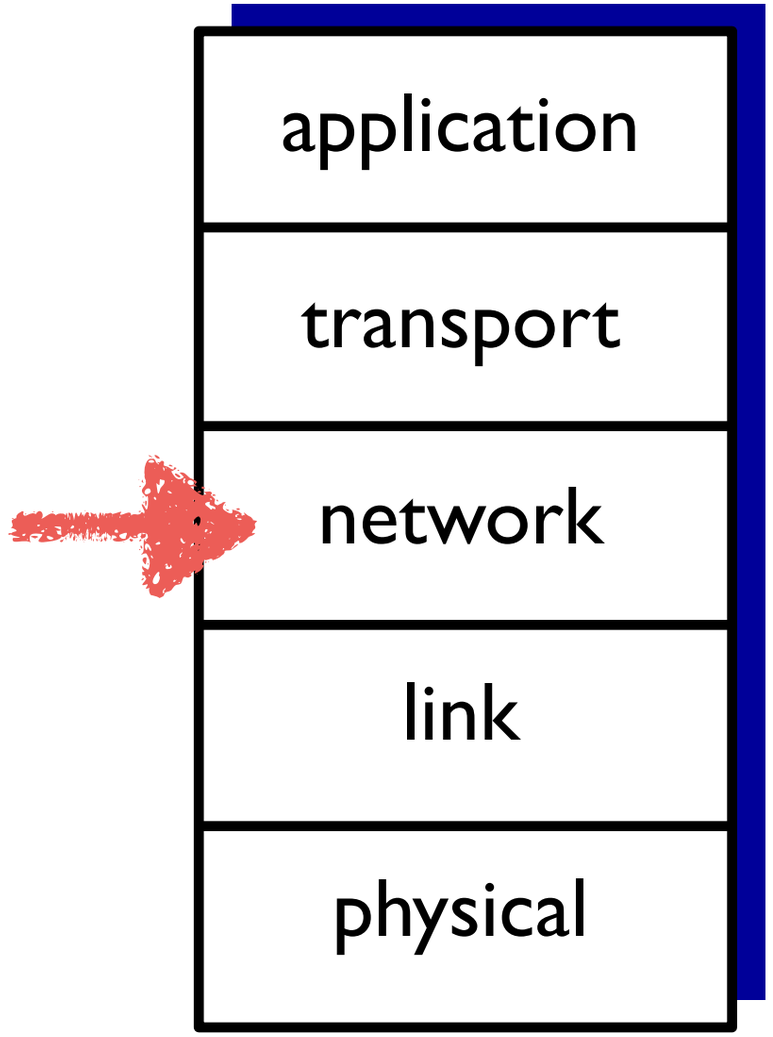
Show your solution to your table

What is good? What can be improved?

Can your client speak to someone else's server?

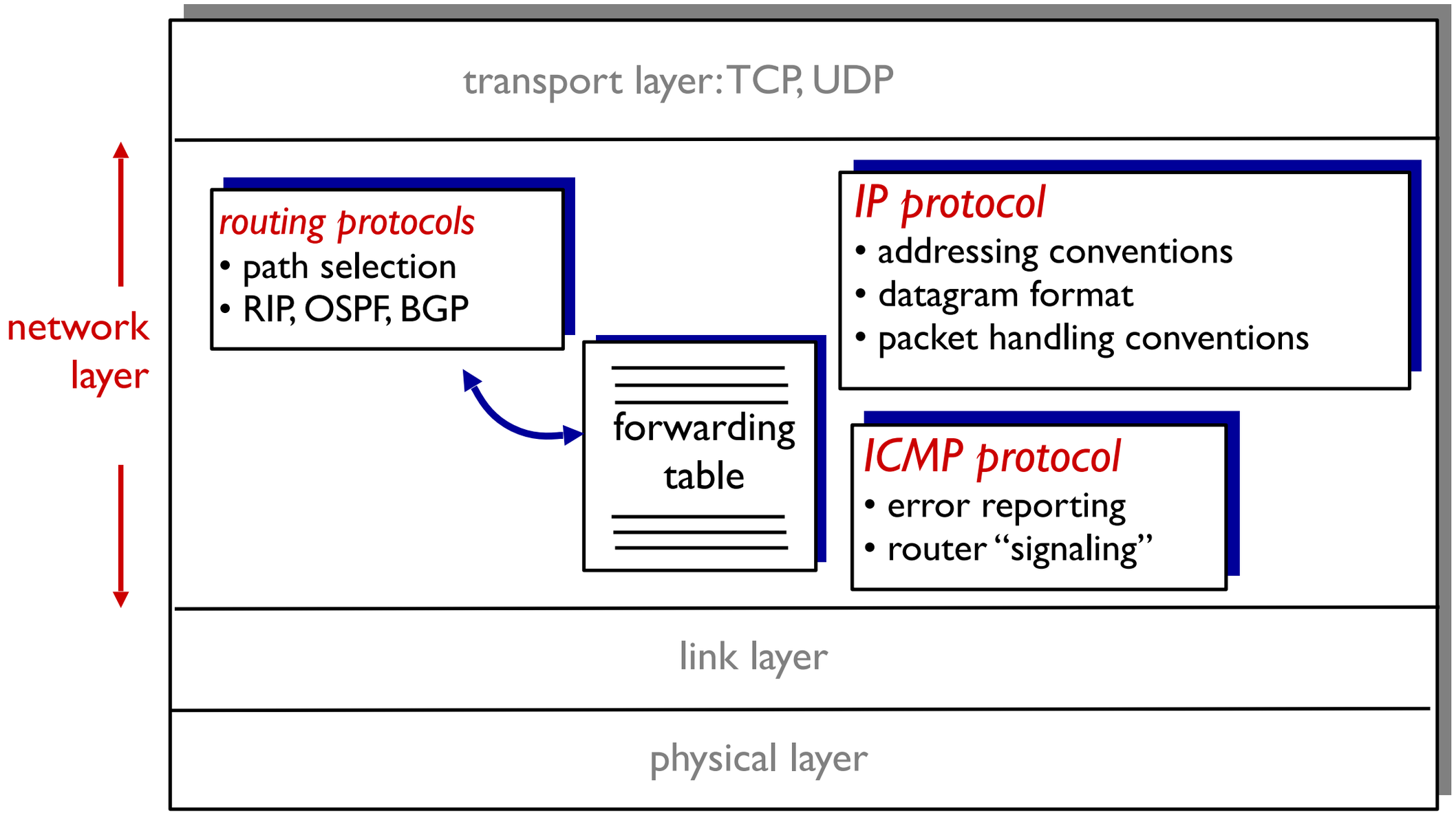
Internet protocol stack

- *application:* supporting network applications
  - FTP, SMTP, HTTP
- *transport:* process-process data transfer
  - TCP, UDP
- *network:* routing of datagrams from source to destination
  - IP, routing protocols
- *link:* data transfer between neighboring network elements
  - Ethernet, 802.111 (WiFi), PPP
- *physical:* bits "on the wire"



The network layer

host, router network layer functions:



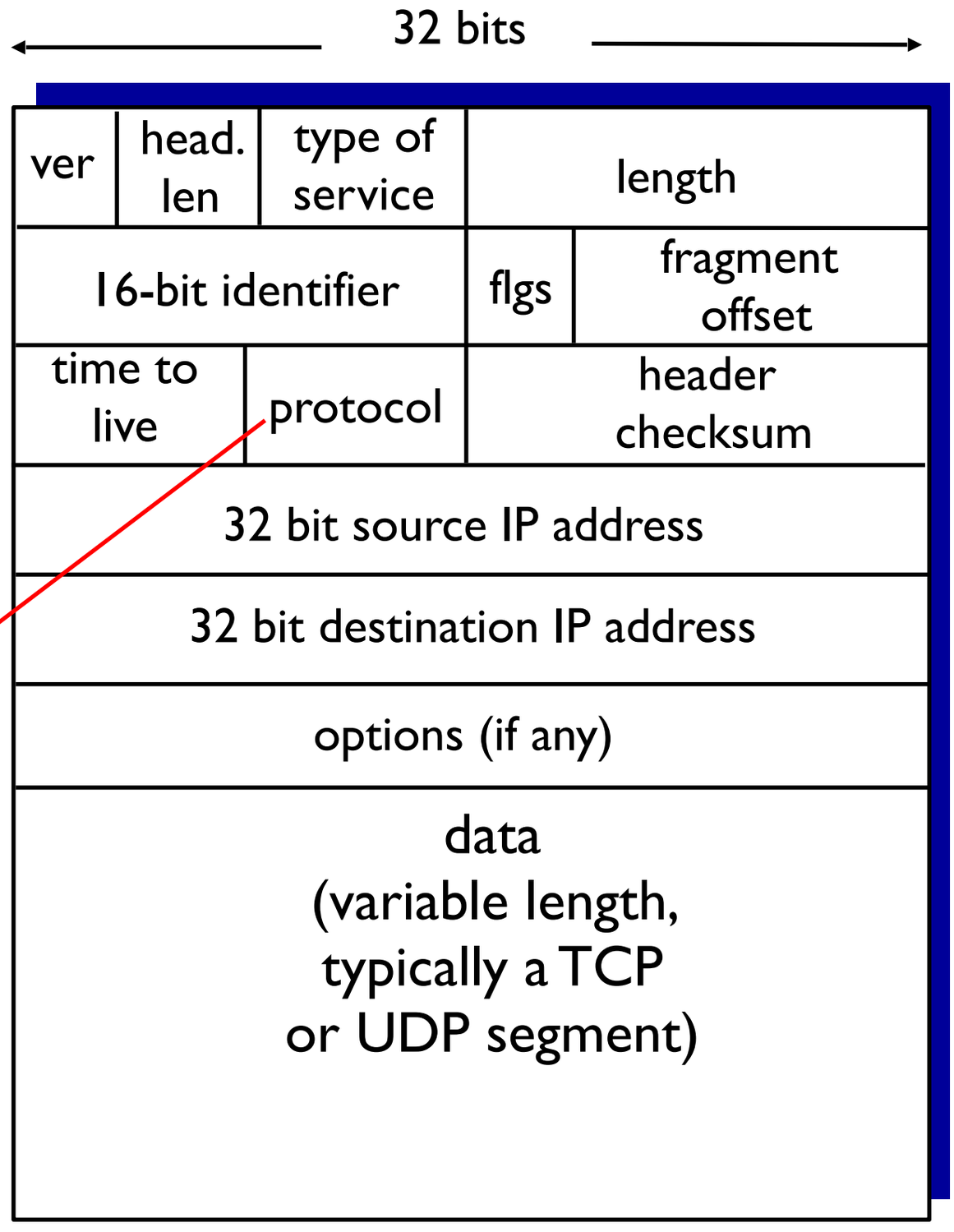
IP datagram format

Network layer

- Determines routing through the network

Source and destination

- IP address

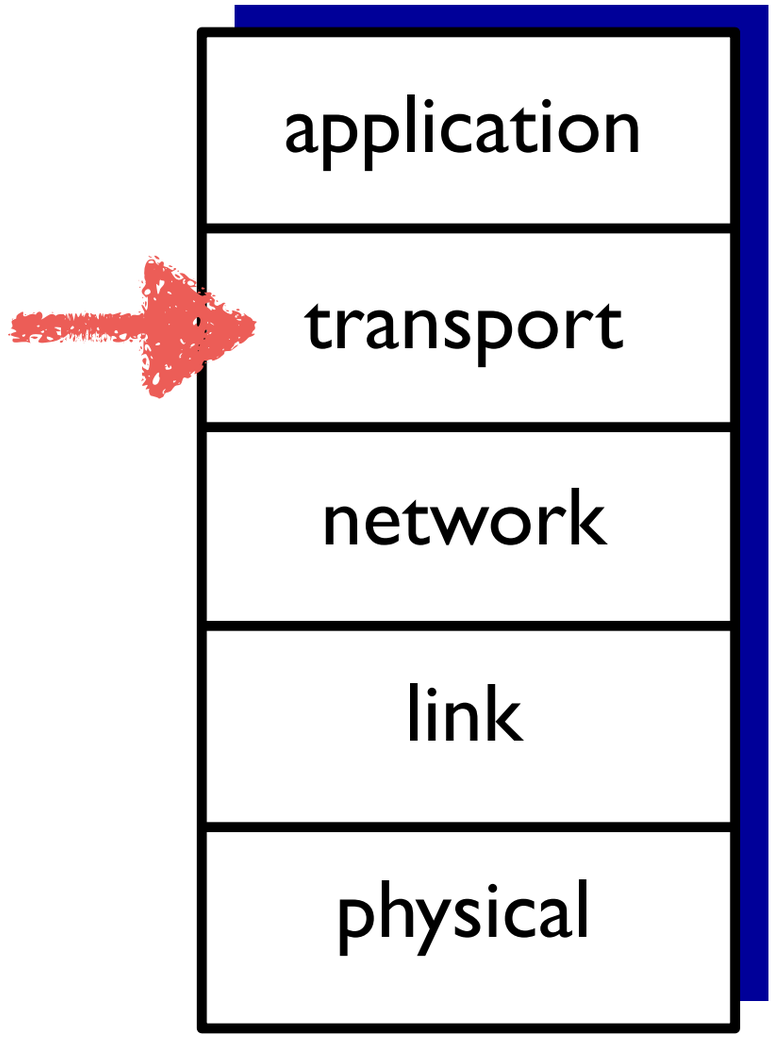


how much overhead?

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead

Transport Layer

How many different **transport protocols** are there?



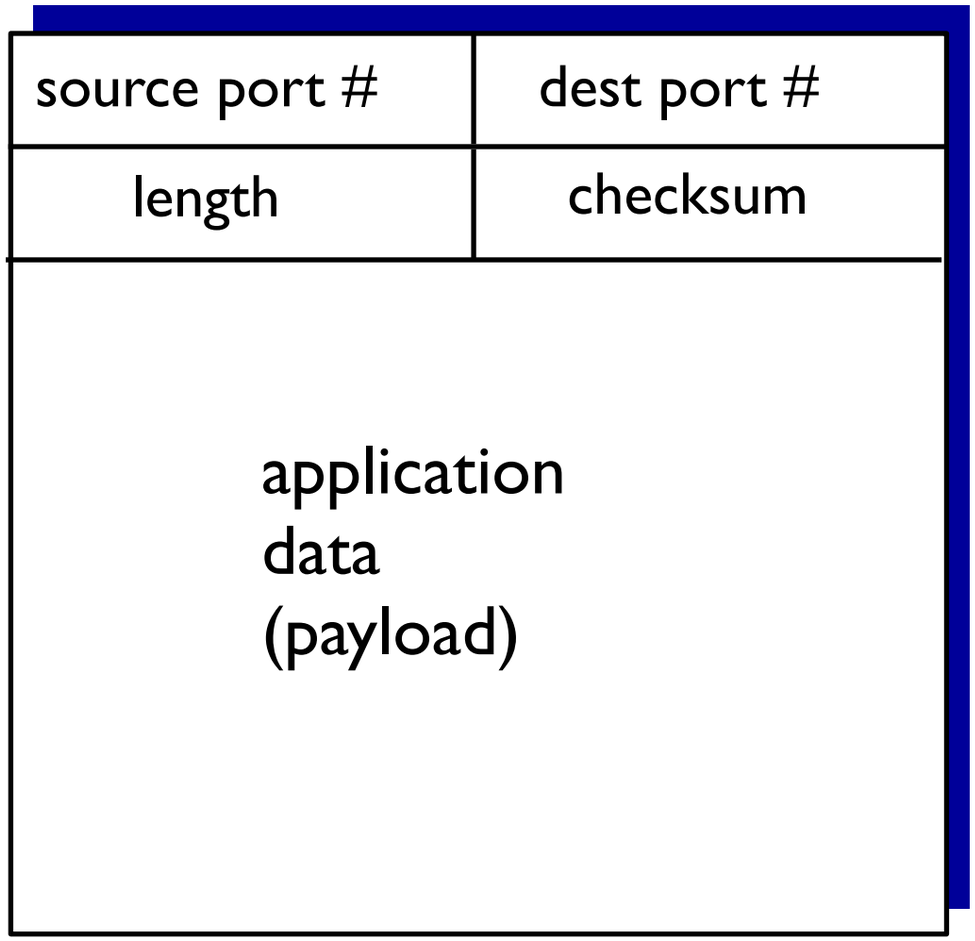
Lots!

https://en.wikipedia.org/wiki/List_of_IP_protocol_numbers

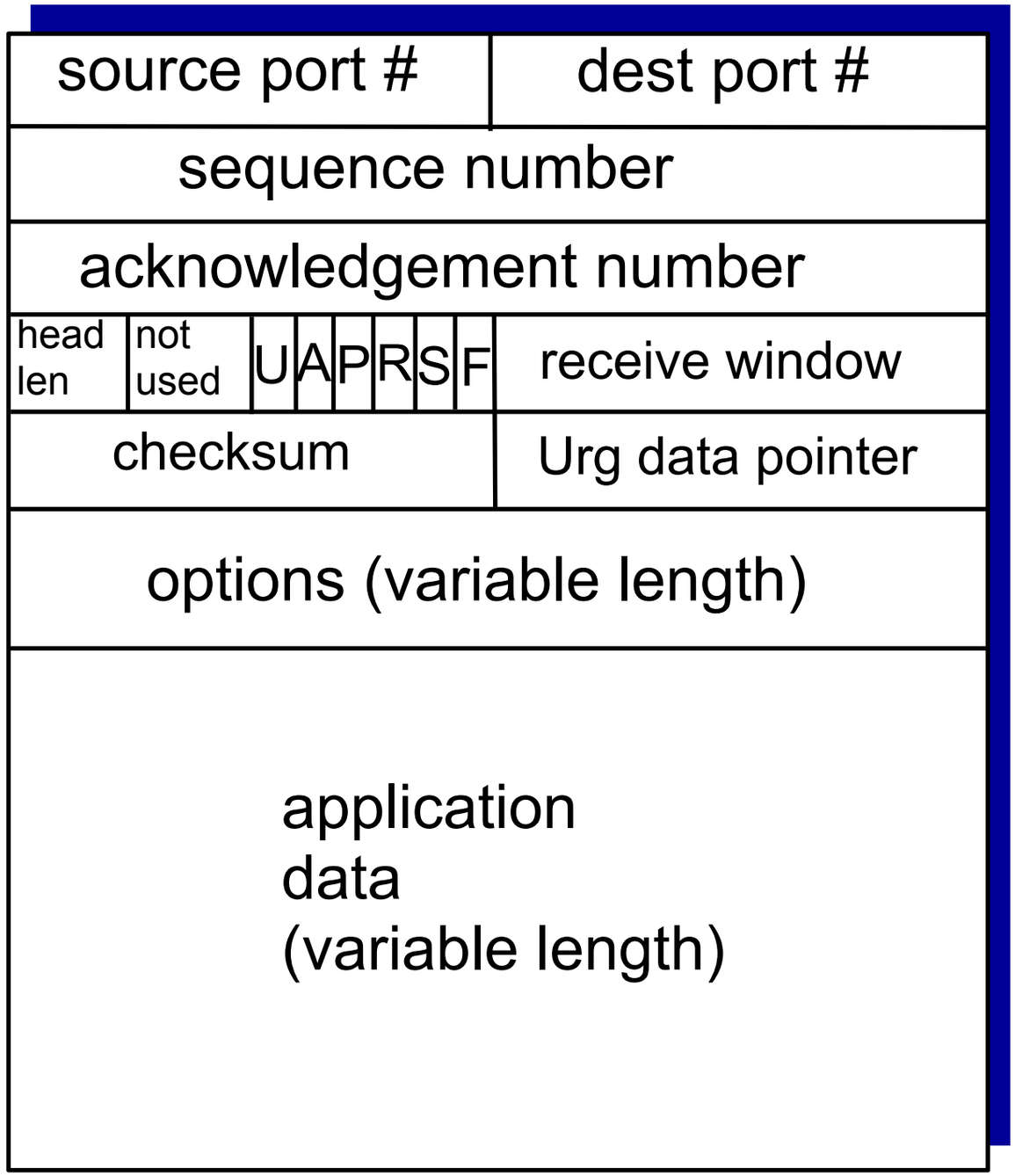
(but less than 256)

Decimal	Hex	Keyword	Protocol	References
0	0x00	HOPOPT	IPv6 Hop-by-Hop Option	RFC 2460
1	0x01	ICMP	Internet Control Message Protocol	RFC 792
2	0x02	IGMP	Internet Group Management Protocol	RFC 1112
3	0x03	GGP	Gateway-to-Gateway Protocol	RFC 823
4	0x04	IP-in-IP	IP in IP (encapsulation)	RFC 2003
5	0x05	ST	Internet Stream Protocol	RFC 1190, RFC 1819
6	0x06	TCP	Transmission Control Protocol	RFC 793
7	0x07	CBT	Core-based trees	RFC 2189
8	0x08	EGP	Exterior Gateway Protocol	RFC 888
9	0x09	IGP	Interior Gateway Protocol (any private interior gateway (used by Cisco for their IGRP))	
10	0x0A	BBN-RCC-MON	BBN RCC Monitoring	
11	0x0B	NVP-II	Network Voice Protocol	RFC 741
12	0x0C	PUP	Xerox PUP	
13	0x0D	ARGUS	ARGUS	
14	0x0E	EMCON	EMCON	
15	0x0F	XNET	Cross Net Debugger	IEN 158
16	0x10	CHAOS	Chaos	
17	0x11	UDP	User Datagram Protocol	RFC 768
18	0x12	MUX	Multiplexing	IEN 90
19	0x13	DCN-MEAS	DCN Measurement Subsystems	
20	0x14	HMP	Host Monitoring Protocol	RFC 869
21	0x15	PRM	Packet Radio Measurement	
22	0x16	XNS-IDP	XEROX NS IDP	
23	0x17	TRUNK-1	Trunk-1	
24	0x18	TRUNK-2	Trunk-2	
25	0x19	LEAF-1	Leaf-1	
26	0x1A	LEAF-2	Leaf-2	
27	0x1B	RDP	Reliable Datagram Protocol	RFC 908
28	0x1C	IRTP	Internet Reliable Transaction Protocol	RFC 938
29	0x1D	ISO-TP4	ISO Transport Protocol Class 4	RFC 905
30	0x1E	NETBLT	Bulk Data Transfer Protocol	RFC 998
31	0x1F	MFE-NSP	MFE Network Services Protocol	
32	0x20	MERIT-INP	MERIT Internodal Protocol	
33	0x21	DCCP	Datagram Congestion Control Protocol	RFC 4340
34	0x22	3PC	Third Party Connect Protocol	
35	0x23	IDPR	Inter-Domain Policy Routing Protocol	RFC 1479
36	0x24	XTP	Xpress Transport Protocol	
37	0x25	DDP	Datagram Delivery Protocol	
38	0x26	IDPR-CMTP	IDPR Control Message Transport Protocol	
39	0x27	TP++	TP++ Transport Protocol	
40	0x28	IL	IL Transport Protocol	
41	0x29	IPv6	IPv6 Encapsulation	RFC 2473
42	0x2A	SDRP	Source Demand Routing Protocol	RFC 1940
43	0x2B	IPv6-Route	Routing Header for IPv6	RFC 2460
44	0x2C	IPv6-Frag	Fragment Header for IPv6	RFC 2460
45	0x2D	IDRP	Inter-Domain Routing Protocol	
46	0x2E	RSVP	Resource Reservation Protocol	RFC 2205
47	0x2F	GRE	Generic Routing Encapsulation	RFC 2784, RFC 2890
48	0x30	MHRP	Mobile Host Routing Protocol	
49	0x31	BNA	BNA	
50	0x32	ESP	Encapsulating Security Payload	RFC 4303
51	0x33	AH	Authentication Header	RFC 4302
52	0x34	I-NLSP	Integrated Net Layer Security Protocol	TUBA
53	0x35	SWIPE	SwIPe	IP with Encryption
54	0x36	NARP	NBMA Address Resolution Protocol	RFC 1735
55	0x37	MOBILE	IP Mobility (Min Encap)	RFC 2004
56	0x38	TLSP	Transport Layer Security Protocol (using Kryptonet key management)	
57	0x39	SKIP	Simple Key-Management for Internet Protocol	RFC 2356
58	0x3A	IPv6-ICMP	ICMP for IPv6	RFC 4443, RFC 4884
59	0x3B	IPv6-NoNxt	No Next Header for IPv6	RFC 2460

TCP vs UDP

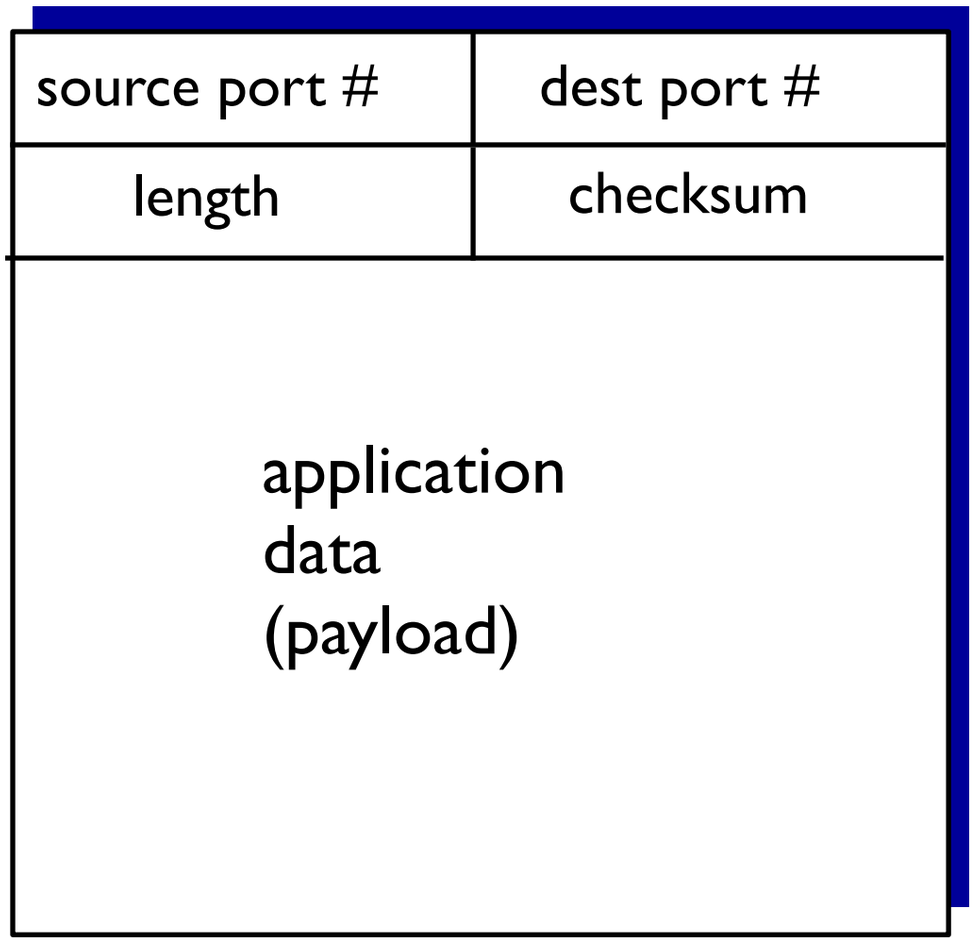


UDP format



TCP format

UDP: Protocol + Data



UDP format

Multiple sockets

- Use port number to differentiate

Data integrity

- Validate checksum

TCP: Protocol + Data

TCP is connection oriented

- 3-way handshake used to establish connection

Ordered transport

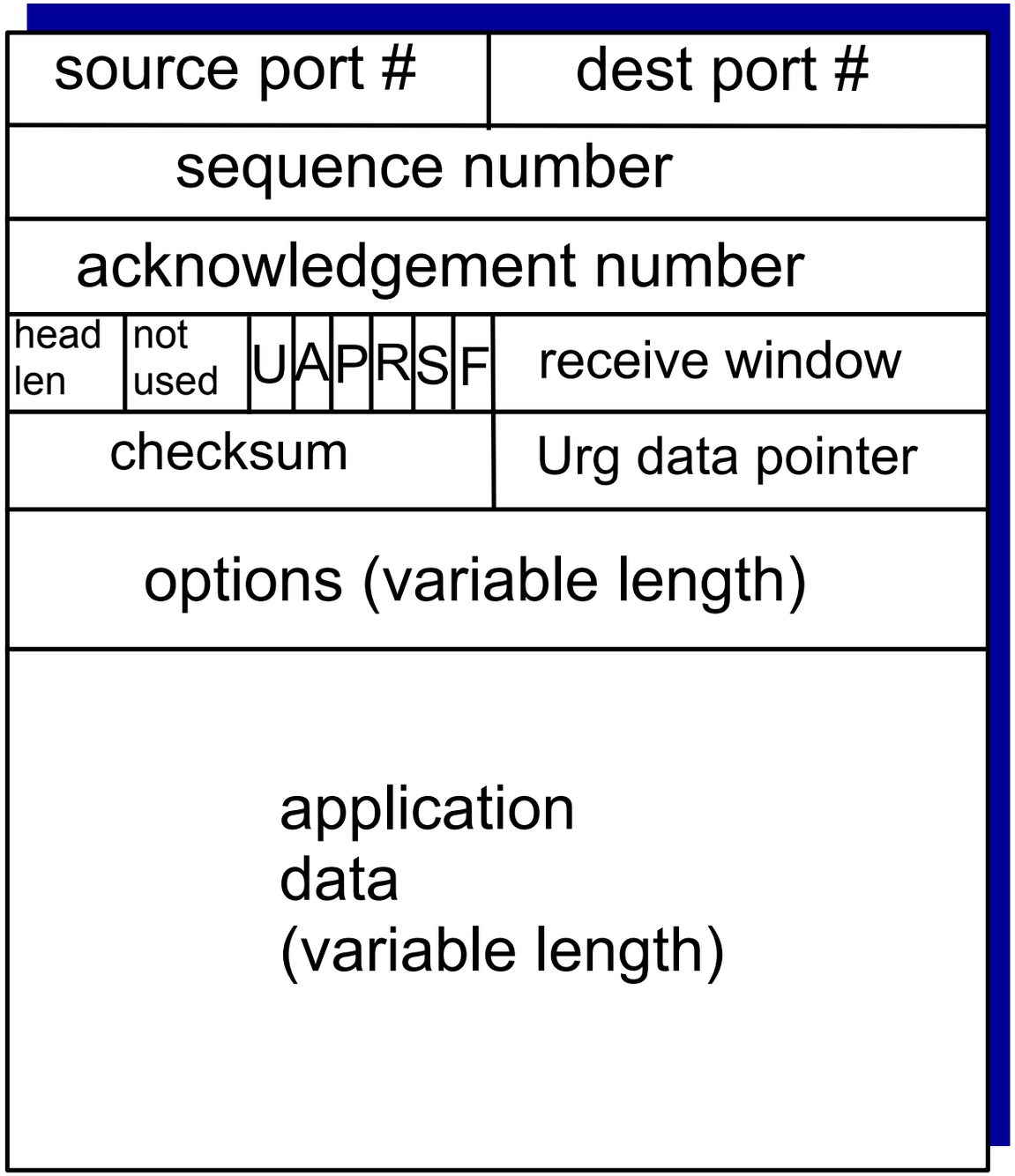
- Sequence number for each packet

Reliable transport

- ACK indicated last sequence number received

Congestion control

- Slow down when packets are dropped



tcp data struct

```
struct tcp_hdr {
 uint16_t src_port; /**< TCP source port. */
 uint16_t dst_port; /**< TCP destination port. */
 uint32_t sent_seq; /**< TX data sequence number. */
 uint32_t recv_ack; /**< RX data acknowledgement sequence num. */
 uint8_t data_off; /**< Data offset. */
 uint8_t tcp_flags; /**< TCP flags */
 uint16_t rx_win; /**< RX flow control window. */
 uint16_t cksum; /**< TCP checksum. */
 uint16_t tcp_urp; /**< TCP urgent pointer, if any. */
}
```

iperf

Use `iperf` to measure network performance

Check the `man` page

try it out:

nimbnode25.seas.gwu.edu

nimbnode27.seas.gwu.edu

iperf

Sample commands:

```
iperf -s

iperf -c nimbnode27.seas.gwu.edu
iperf -c 10.1.1.27
iperf -c 10.1.1.27 -u -b 500M
iperf -c 10.1.1.27 -u -b 1500M
iperf -c 10.1.1.27 -u -b 1500M -l 256
```

Network Performance

Which is faster? TCP or UDP?

What affects speed?

Packet Rates

Ethernet packet size: 46 to 1500 bytes

```
10Gbps = 10,000,000,000 bits per second
       =  1,250,000,000 bytes per second

       =  27,173,913 tiny pkts per second
       =  833,333 large pkts per second
```

(approximately)

What work has to happen for each packet?

Tracing Program Execution

How can we figure out what our program is doing?

`ltrace` - library tracer

`strace` - system call tracer

Look at the `man` page for each

Try them each out on some different programs

- from HelloWorld to TrafficGen

Packet Path

Why do we have layers?

???

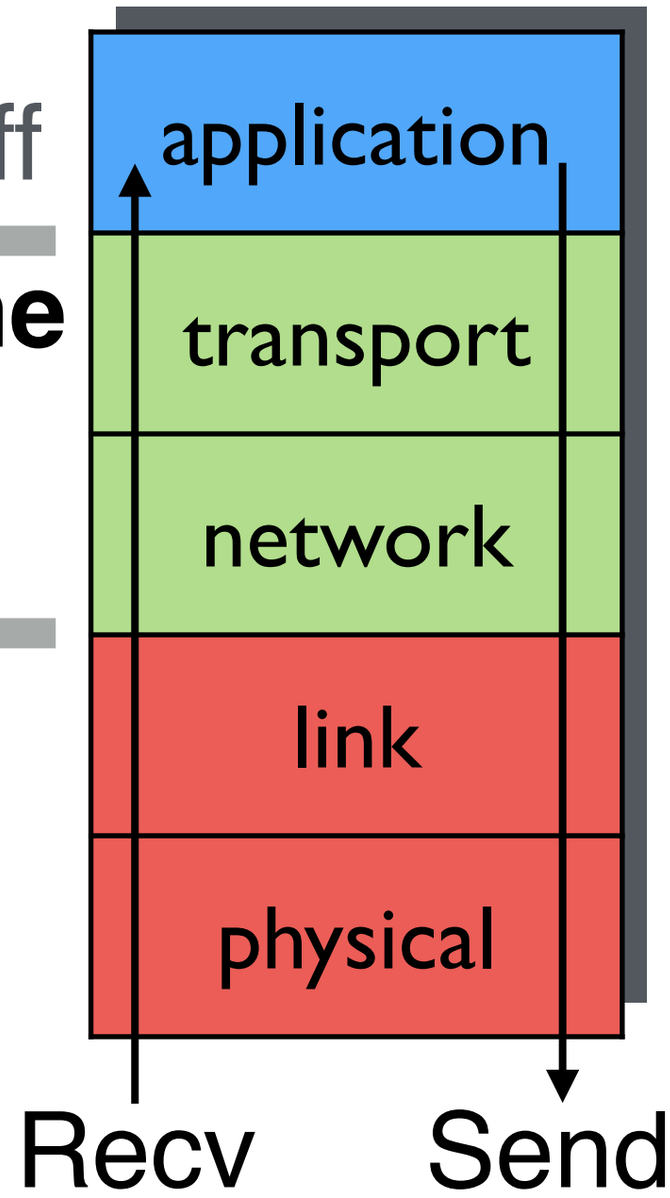
Why are there multiple layers dealt with by the NIC and OS?

???

App Stuff

Awesome OS Stuff

Boring ECE Stuff



Linux kernel map

2.6.36

functionalities
layers

human interface | system | processing | memory | storage | networking

user space interfaces
system calls and system files

HI char devices
cdev_add
input_fops
video_fops
snd_fops
console_fops
sys_capset
fb_fops
sys_syslog

interfaces core
kernel/
System Call Interface
system files
linux/syscalls.h
linux/uaccess.h
copy_from_user
/proc /sysfs /dev
sysfs_ops
register_chrdev
sys_ioctl
cdev_add
cdev_map
sys_epoll_create
cdev
sys_reboot
/sys/class/
sys_init_module
kvm_dev_ioctl

processes
kernel/
fs/exec.c
kernel/signal.c
sys_kill
sys_signal
do_sigaction
sys_fork
sys_vfork
sys_clone
sys_execve
linux_binfmt
sys_gettimeofday
sys_time
sys_times
sys_futex
sys_nanosleep

memory access
mm/
sys_brk
sys_mmap
shm_vm_ops
sys_shmctl
sys_shmat
sys_sysinfo
si_meminfo
sys_mprotect
sys_mincore
/dev/mem
mem_fops
mmap_mem
/proc/meminfo
/proc/self/maps
sys_msync

files & directories access
fs/
sys_chroot
sys_pivot_root
sys_mkdir
sys_chdir
sys_open
sys_read
sys_tee
sys_poll
sys_pipe
sys_select
sys_flock
sys_write
do_path_lookup
sys_getdents
sys_readv
sys_splice
sys_sendfile
sys_swapon
iovec
alloc_file
sys_chown
sys_chmod
notify_change
fd
sys_newfstat
sys_fsync
fsnotify_change
fget
sys_mount
inode_setattr
sys_sync
sys_sysfs

sockets access
net/
sys_socketcall
sys_connect
sys_accept
sys_bind
sys_listen
sys_sendmsg
sys_recvmsg
sys_setsockopt
sys_socket
/proc/net/
tcp4_seq_show
sg_proc_seq_show_dev
rt_cache_seq_show
sock_ioctl

virtual

security
security/
linux/security.h
security_capset
may_open
security_socket_create
inode_permission
security_inode_create
security_ops
selinux_ops

Device Model
drivers/base/
driver_init
kobject
linux/kobject.h
bus_register
kset
linux/device.h
bus_type
device_create
device
class
device_type
driver_register
device_driver
probe
kvm

threads
schedule_work
create_workqueue
INIT_WORK
queue_work
work_struct
workqueue_struct
kthread_create
kernel_thread
current
thread_info
do_fork

virtual memory
vmalloc_init
find_vma_prepare
vmalloc
vfree
vmlist
vm_struct
virt_to_page

Virtual File System
file
vfs_fsync
file_systems
vfs_fstat
vfs_read
vfs_write
vfs_getattr
vfs_create
inode
fsnotify
inode_operations
file_operations
file_system_type
get_sb
super_block
do_splice_direct
generic_file_aio_read
ramfs_fs_type

protocol families
inet_init
__sock_create
socket
inet_family_ops
inet_create
unix_family_ops
proto_ops
inet_dgram_ops
inet_stream_ops
socket_file_ops

bridges
cross-functional modules

debugging
sys_ptrace
log_buf
register_kprobe
printk
handle_sysrq
oprofile_start
kgdb_breakpoint
oprofile_init

synchronization
lock_kernel
kernel_flag
mutex_lock_interruptible
mutex_unlock
mutex
owner
wait_for_completion
complete
add_timer
timer_list
run_timer_softirq
down_interruptible
up
semaphore
wait_event
wake_up
msleep
spin_lock_irqsave
spin_unlock_irqrestore

memory mapping
mm/mmap.c
do_mmap
do_mmap_pgoff
kmem_cache_alloc
vma_link
mm_struct
vm_area_struct

page cache
address_space
bdi_writeback_thread
do_writepages

swap
si_swapinfo
swap_info
kswapd
do_swap_page
wakeup_kswapd

networking storage
nfs_file_operations
smb_fs_type
cifs_file_ops
iscsi_tcp_transport

socket splice
sock_sendpage
tcp_sendpage
udp_sendpage
sock_splice_read
tcp_splice_read

logical
functions implementations

HI subsystems
oss
alsa
video_device
mousedev_handler
tty

system run
init/
kernel/
boot, shutdown
power management
init/main.c
start_kernel
do_initcalls
mount_root
run_init_process
kernel_restart
kernel_power_off
hibernate
machine_ops

Scheduler
kernel/sched.c
task_struct
schedule_timeout
schedule
setup_timer
process_timeout
activate_task
rq
context_switch

logical memory
physically mapped memory
mm_init
kmalloc
kfree
pgd_t
pmd_t
pte_t

logical file systems
ext4_file_operations
ext4_get_sb
ext4_readdir

protocols
proto
/proc/net/protocols
udp_prot
tcp_prot
udp_recvmsg
udp_sendmsg
tcp_recvmsg
tcp_sendmsg
tcp_v4_rcv
udp_rcv
tcp_transmit_skb
NF_HOOK
nf_hooks
ip_queue_xmit
ip_local_deliver
ip_push_pending_frames
ip_route_input
alloc_skb
ip_rcv
sk_buff
ip_output

device control

drivers/

abstract devices and HID class drivers
drivers/input/
drivers/media/
sound/
console
kbd
fb_ops
mousedev
drm_driver

generic HW access
request_region
pci_driver
request_mem_region
pci_register_driver
pci_request_regions
usb_driver
usb_submit_urb
usb_hcd_giveback_urb
ioremap
usb_hcd

interrupts core
request_irq
tasklet_struct
jiffies_64++
tasklet_action
do_timer
setup_irq
tick_periodic
timer_interrupt
do_softirq
do_IRQ
irq_desc
softirq_init
include/asm/
arch/x86/

Page Allocator
/proc/slabinfo
mm/slob.c
mm/slub.c
mm/slab.c
__free_pages
kmem_cache
kmem_cache_init
kmem_cache_alloc
__free_one_page
__get_free_pages
page
__alloc_pages
vm_stat
totalram_pages
try_to_free_pages

block devices
block/
gendisk
block_device_operations
request_queue
init_scsi
scsi_device
scsi_driver
sd_fops
usb_storage_driver
usb_stor_host_template

network interface
linux/netdevice.h
netif_receive_skb
netif_rx
dev_queue_xmit
register_netdev
net_device
dev_ioctl
alloc_netdev_mq
ieee80211_alloc_hw
ether_setup
ieee80211_rx
netif_carrier_on
ieee80211_xmit
drivers/net/

hardware interfaces
drivers, registers and interrupts

HI peripherals device drivers
uvc_driver
vga_con
ac97_driver
atkbd_drv
i8042_driver
psmouse
drivers/media/video/

device access and bus drivers
ehci_irq
writew
readw
ehci_urb_enqueue
outw
inw
usb_hcd_irq
pci_read
pci_write

CPU specific
arch/x86/
setup_arch
x86_init
trap_init
early_trap_init
start_thread
native_init_IRQ
switch_to
/proc/interrupts
set_intr_gate
system_call
show_regs
interrupt
pt_regs
atomic_t
cli
sti

physical memory operations
arch/x86/mm/
mem_init
get_page_from_freelist
zonelist
zone
free_area
free_list
out_of_memory
die
num_physpages
do_page_fault

disk controller drivers
scsi_host_alloc
libata
Scsi_Host
ahci_pci_driver
aic94xx_init

network device drivers
usbnet_probe
ipw2100_pci_init_one
zd1201_probe
e1000_xmit_frame
e1000_intr

electronics

user peripherals
keyboard
camera
mouse
graphics card
audio

I/O
I/O mem
PCI controller
I/O ports
ACPI
USB controller

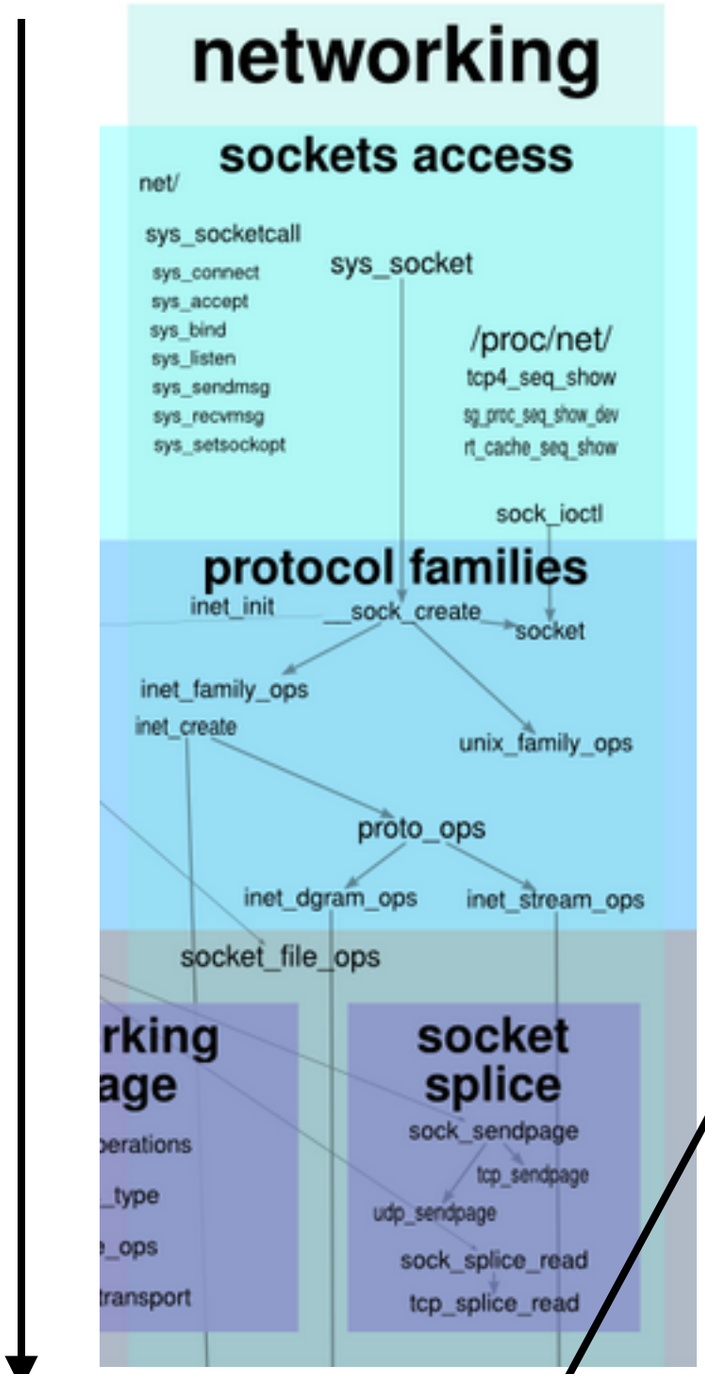
CPU
registers
APIC
interrupt controller

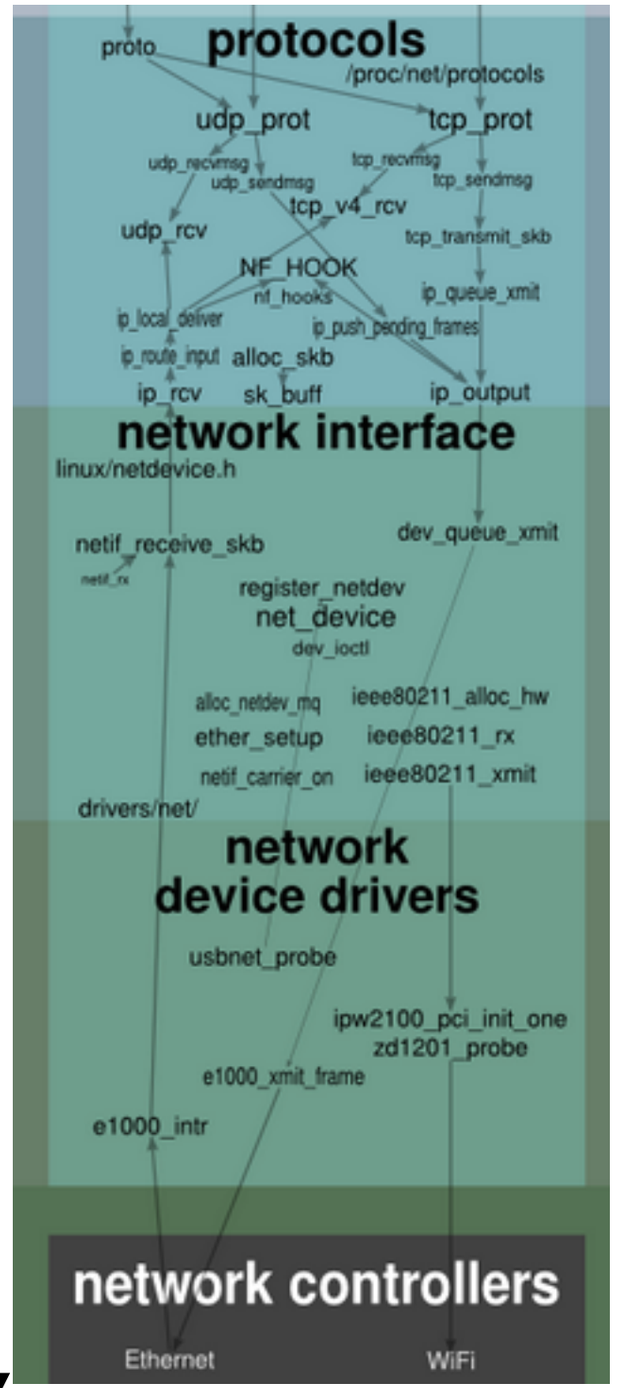
memory
RAM
DMA
MMU

disk controllers
SCSI
SATA

network controllers
Ethernet
WiFi

Linux Network Stack





Receiving a Packet

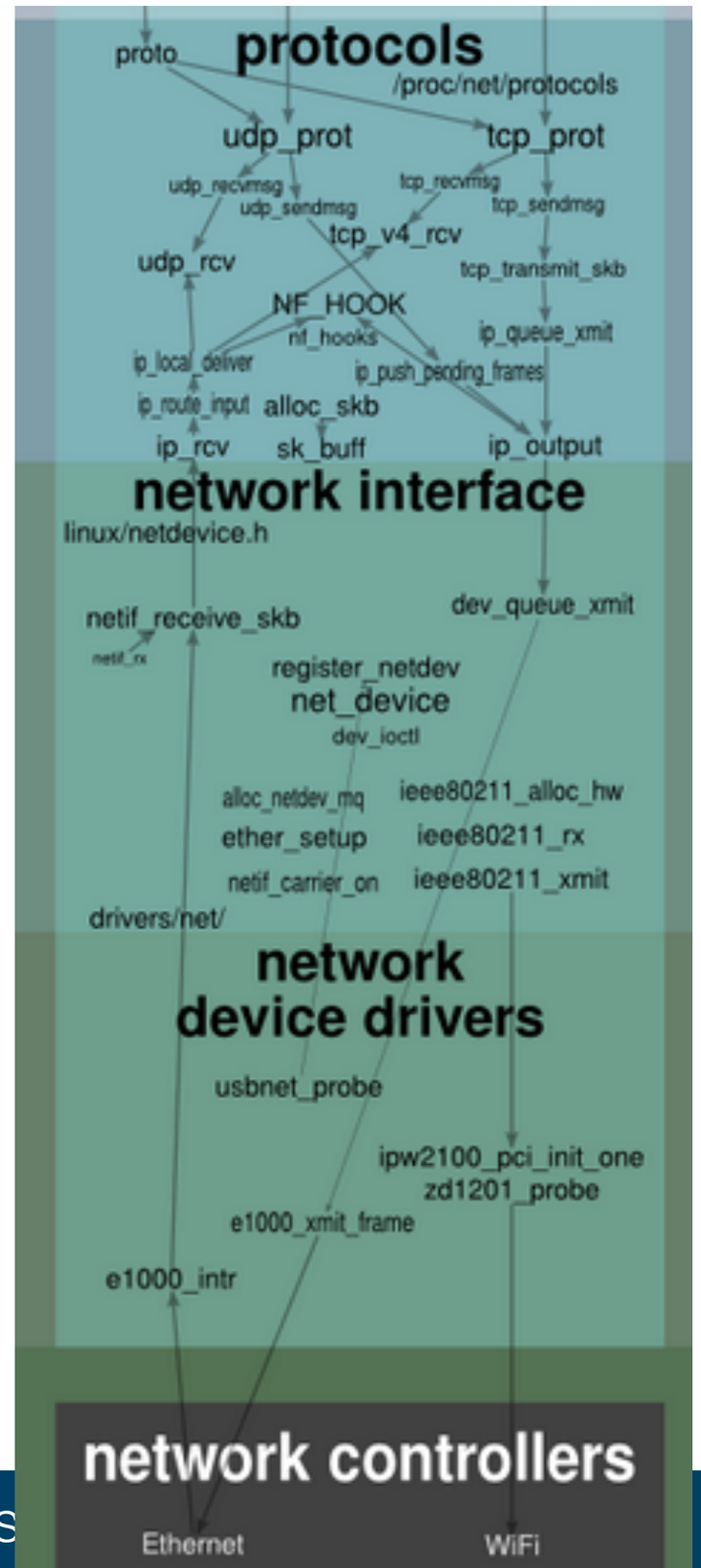
NIC registers an interrupt handler with the OS

- Handler is called when packet arrives
- Packet is copied into kernel memory

Handler uses an **sk_buff** to refer to the packet

Calls **ip_recv()**

- processes IP header
- Determines if packet is local or to be forwarded



Receiving a Packet

Determine packet's transport protocol

- UDP, TCP, etc

Match the UDP/TCP packet to the correct socket

- Use destination port number

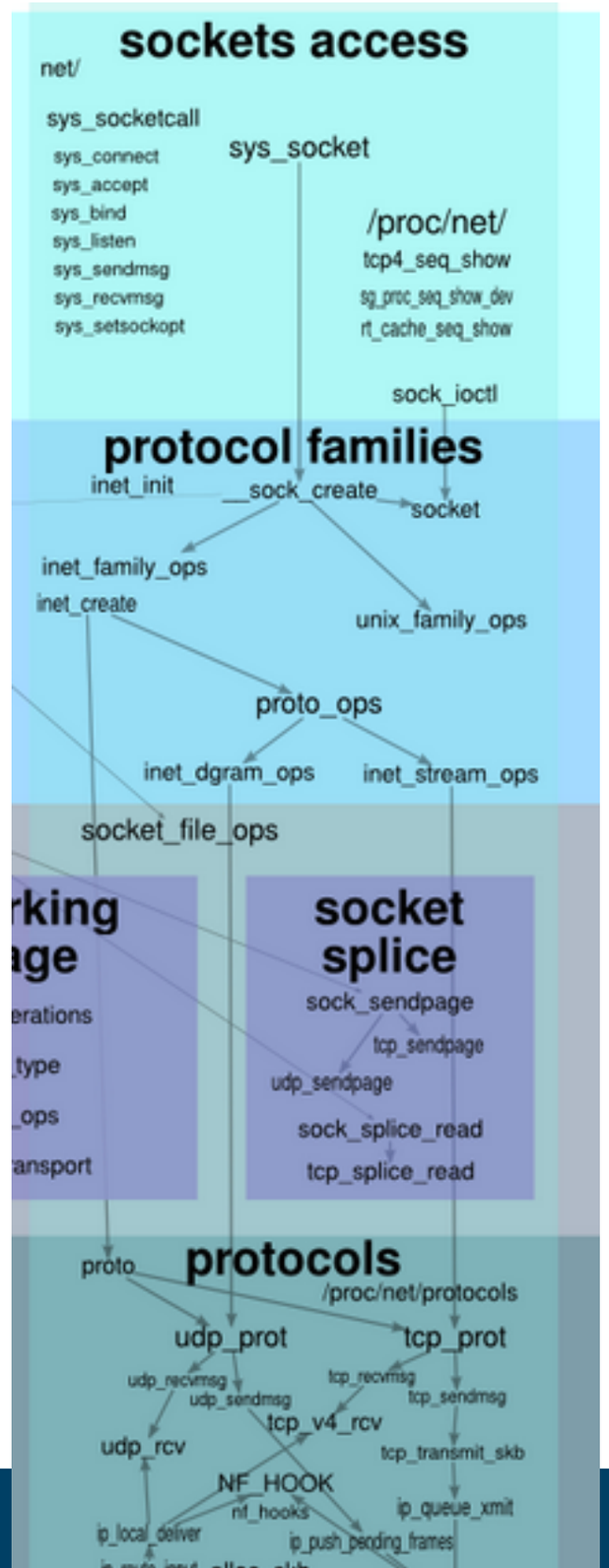
Hold onto the packet

- until sometime later…

User app calls **recv()**

- Kernel calls **copy_to_user()**

Data is available to application



Keep Git Clean

Follow the style guide

- Formatting, code structure, etc

Focused Pull Requests

- a PR should only have code that fixes a single Issue

Up to date forks

- You need to pull from the origin repo to keep your fork current

Repo Rules

Work in your fork

Keep your master branch clean

- Never directly modify your master

Do your work in branches in your fork

- Naming scheme **i<issue#>-<short-name>**
  - i25-add-udp
- Only make edits to that branch related to the issue

Push your branch to your fork

Create Pull Request between your branch and origin

Regularly pull from origin into your master