CSCI-GA.2250-001

Operating Systems

Networking

Hubertus Franke
frankeh@cs.nyu.edu

TCP/IP protocol family

- IP : Internet Protocol
  - UDP : User Datagram Protocol
    - RTP, traceroute
  - TCP : Transmission Control Protocol
    - HTTP, FTP, ssh

What is an internet?

- A set of *inter*connected *net*works
- The **I**nternet is the most famous example

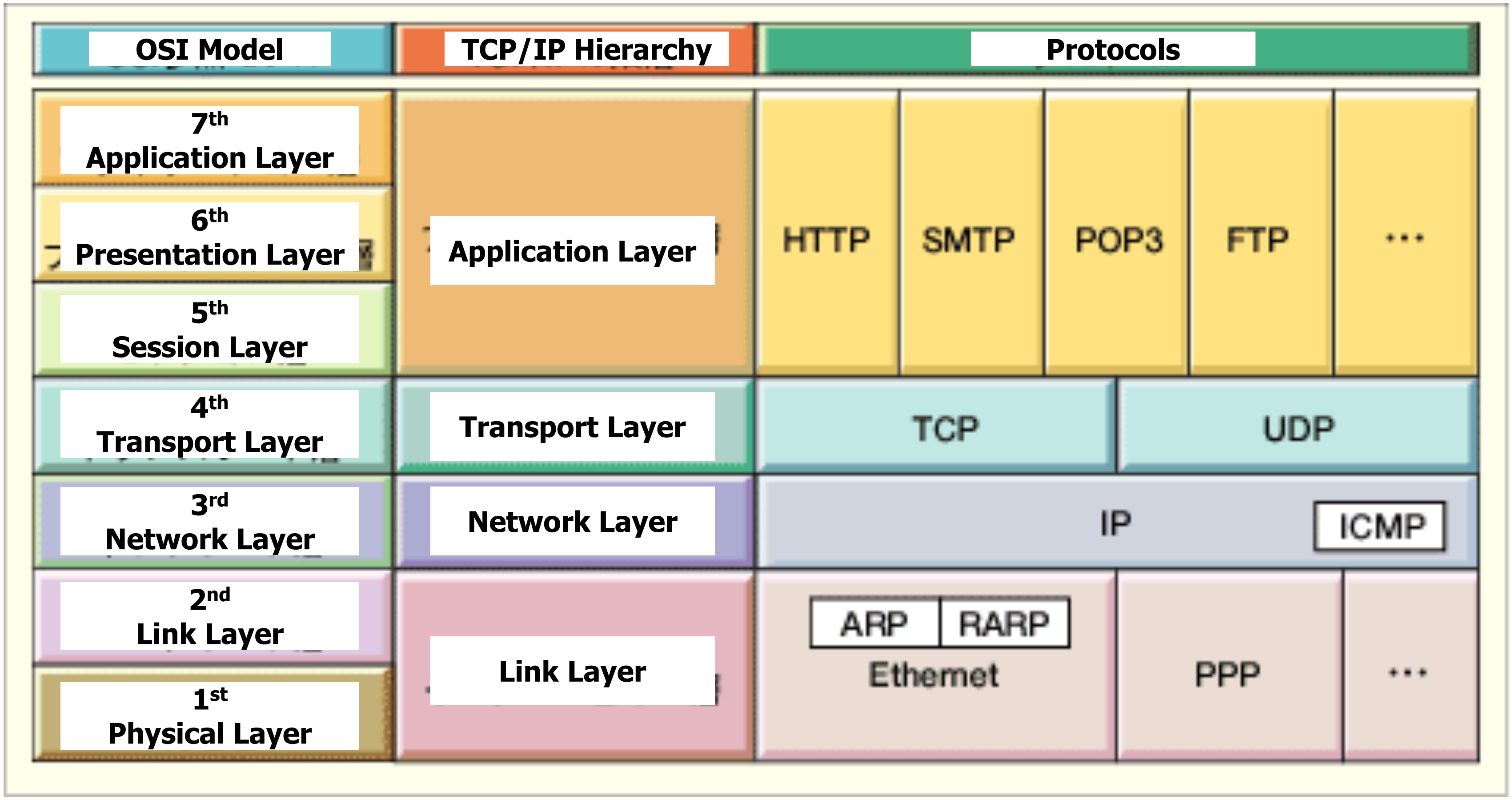
- Networks can be completely different
  - Ethernet, ATM, modem, …
  - (TCP/)IP is what links them

What is an internet? (cont)

- *Routers* (nodes) are devices on multiple networks that pass traffic between them
- Individual networks pass traffic from one router or endpoint to another
- TCP/IP hides the details as much as possible

OSI: Open Systems Interconnect

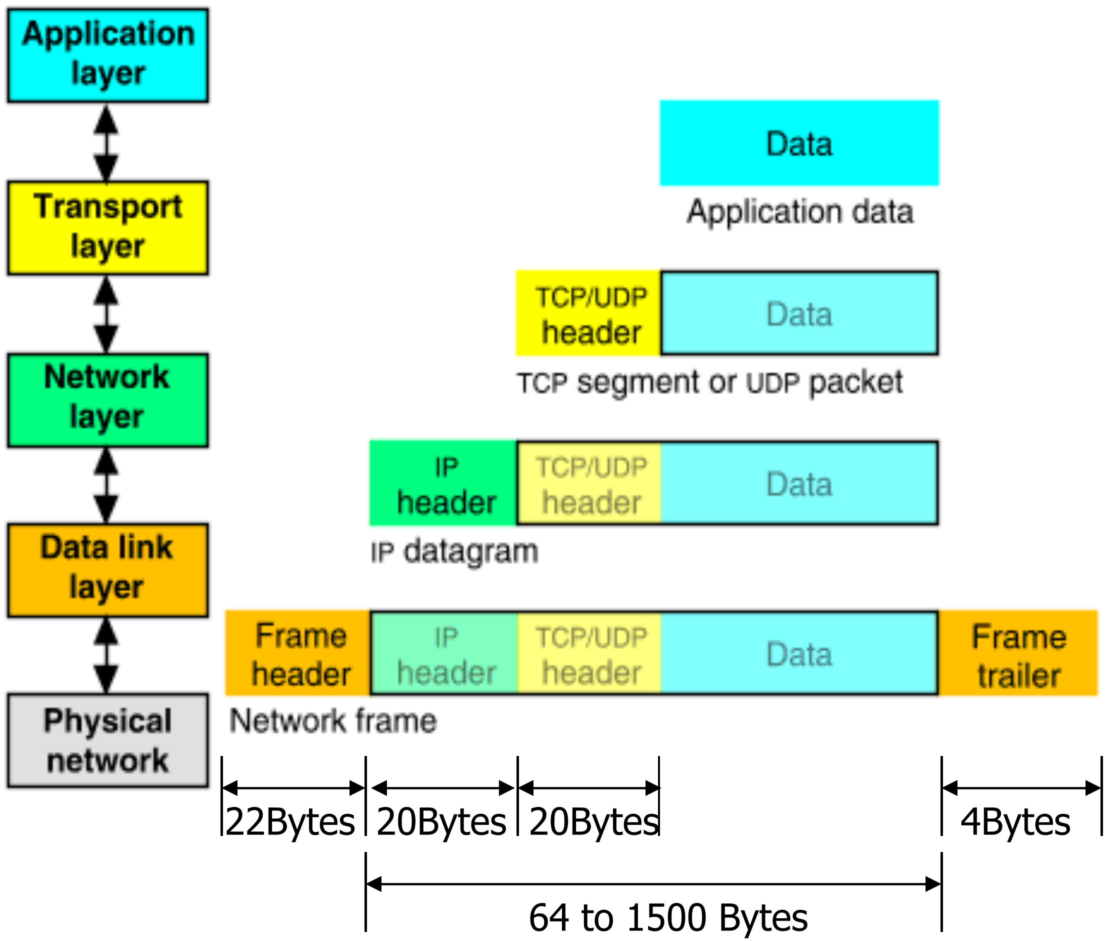
OSI and Protocol Stack

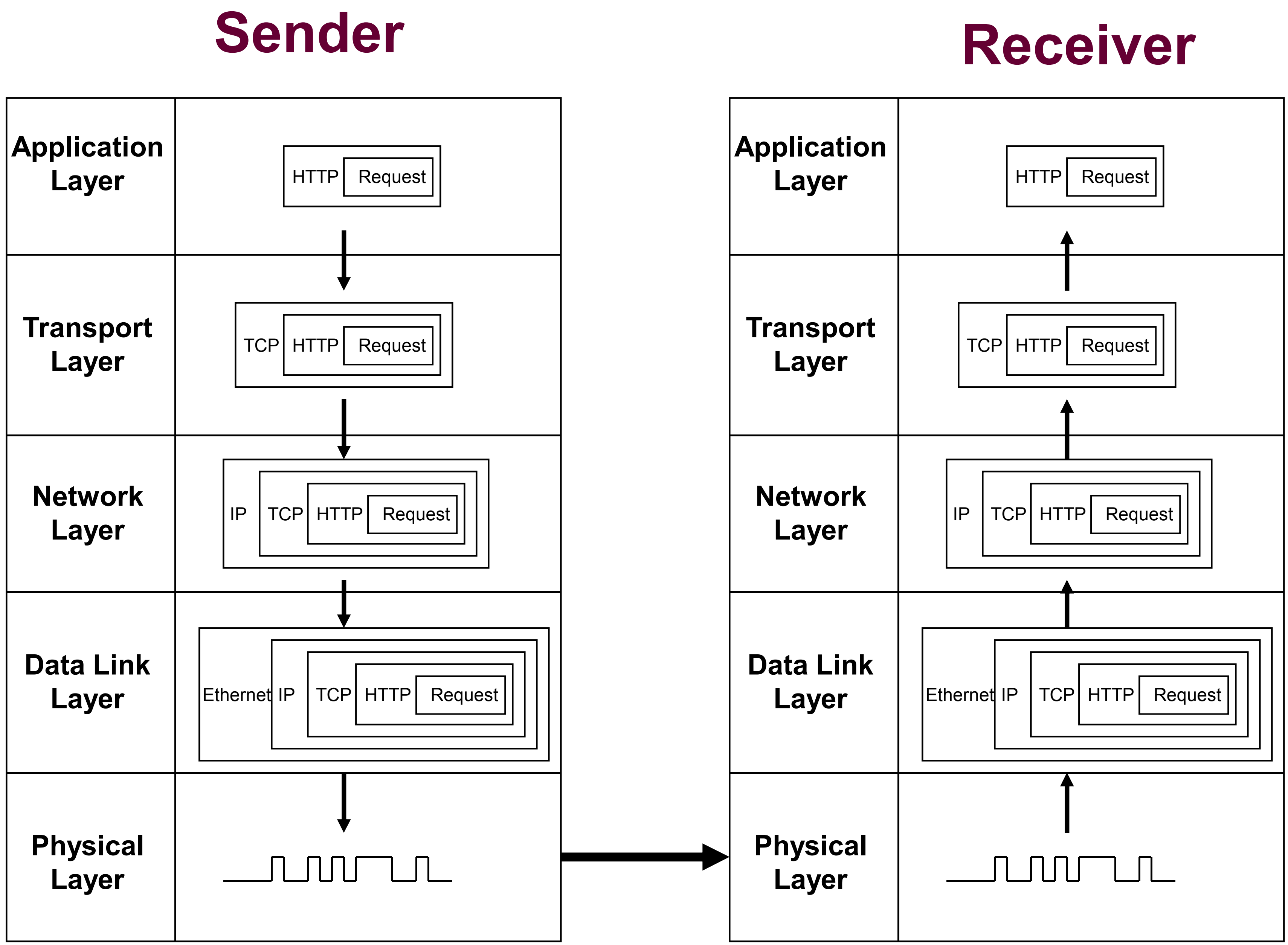


Link Layer : includes device driver and network interface card
Network Layer : handles the movement of packets, i.e. Routing
Transport Layer : provides a reliable flow of data between two hosts
Application Layer : handles the details of the particular application

Packet Encapsulation

- The data is sent down the protocol stack
- Each layer adds to the data by prepending headers




Sender
Receiver
Application Layer
HTTP
Request
Transport Layer
TCP
HTTP
Request
Network Layer
IP
TCP
HTTP
Request
Data Link Layer
Ethernet
IP
TCP
HTTP
Request
Physical Layer
Application Layer
HTTP
Request
Transport Layer
TCP
HTTP
Request
Network Layer
IP
TCP
HTTP
Request
Data Link Layer
Ethernet
IP
TCP
HTTP
Request
Physical Layer

IP

- Responsible for end to end transmission
- Sends data in individual packets
- Maximum size of packet is determined by the networks
  - Fragmented if too large
- Unreliable
  - Packets might be lost, corrupted, duplicated, delivered out of order

IP addresses

- 4 bytes
  - e.g. 163.1.125.98
  - Each device normally gets one (or more)
  - In theory there are about 4 billion available
- But…

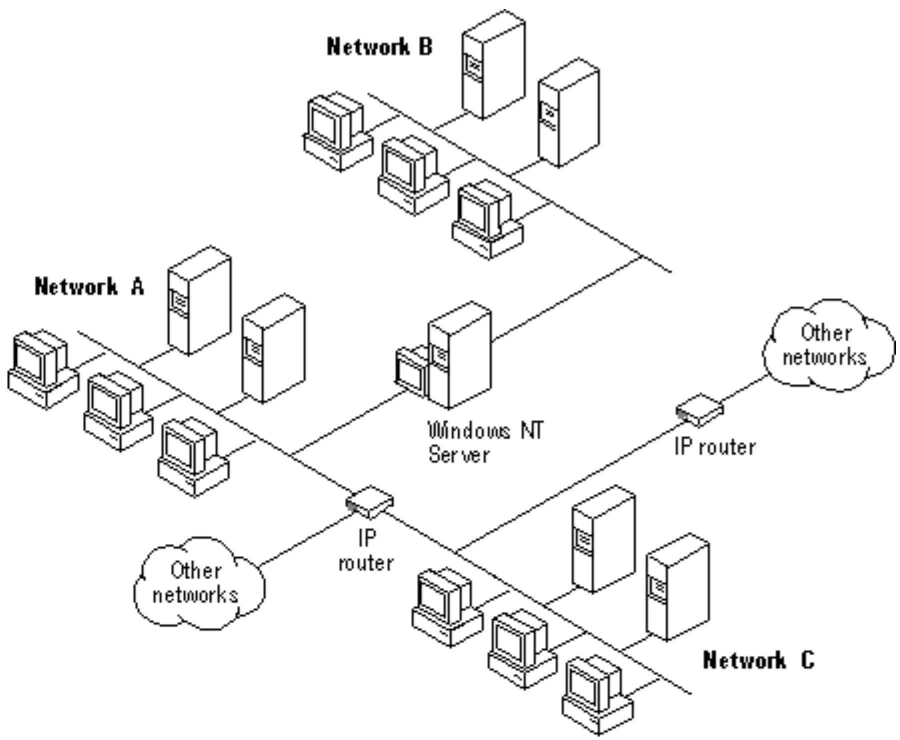
Routing

- How does a device know where to send a packet?
  - All devices need to know what IP addresses are on directly attached networks
  - If the destination is on a local network, send it directly there

Routing (cont)

- If the destination address isn't local
  - Most non-router devices just send everything to a single local router
  - Routers need to know which network corresponds to each possible IP address

Routing


Network B
Network A
Windows NT
Server
Other
networks
IP router
IP
router
Other
networks
Network C

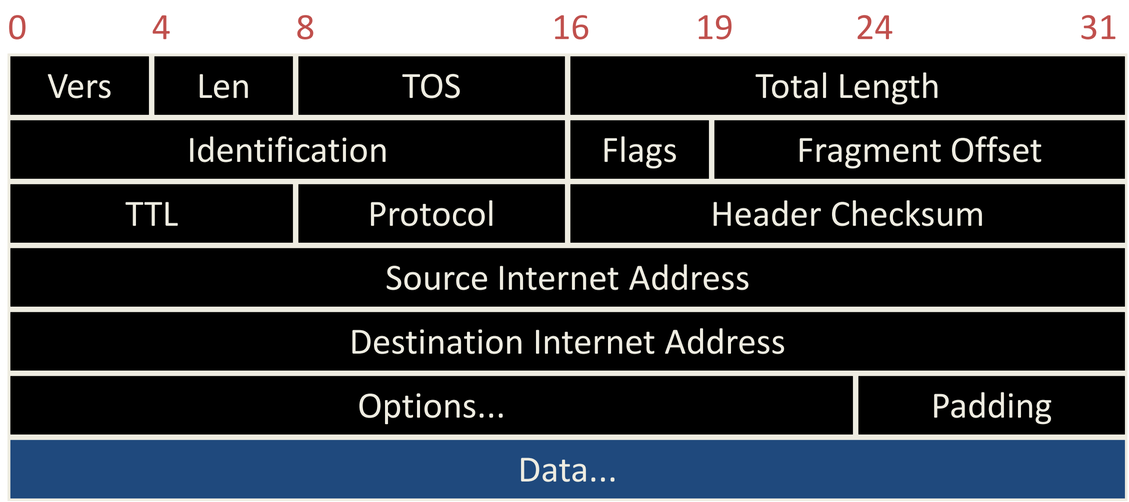
Allocation of addresses

- Controlled centrally by ICANN
  - Fairly strict rules on further delegation to avoid wastage
    - Have to demonstrate actual need for them
- Organizations that got in early have bigger allocations than they really need

IP packets

- Source and destination addresses
- Protocol number
  - 1 = ICMP, 6 = TCP, 17 = UDP
- Various options
  - e.g. to control fragmentation
- Time to live (TTL)
  - Prevent routing loops

IP Datagram

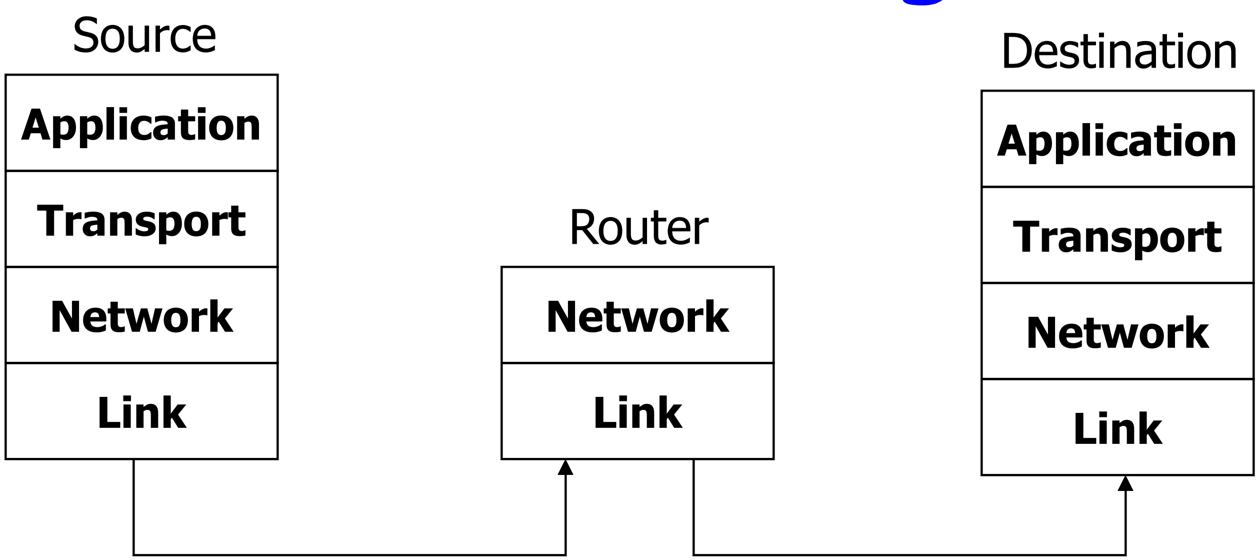


Field	Purpose
Vers	IP version number
Len	Length of IP header (4 octet units)
TOS	Type of Service
T. Length	Length of entire datagram (octets)
Ident.	IP datagram ID (for frag/reassembly)
Flags	Don't/More fragments
Frag Off	Fragment Offset

Field	Purpose
TTL	Time To Live - Max # of hops
Protocol	Higher level protocol (1=ICMP, 6=TCP, 17=UDP)
Checksum	Checksum for the IP header
Source IA	Originator's Internet Address
Dest. IA	Final Destination Internet Address
Options	Source route, time stamp, etc.
Data...	Higher level protocol data

We only looked at the IP addresses, TTL and protocol #

IP Routing



- Routing Table
  - Destination IP address
  - IP address of a next-hop router
  - Flags
  - Network interface specification

UDP

- Thin layer on top of IP
- Adds packet length + checksum
  - Guard against corrupted packets
- Also source and destination *ports*
  - Ports are used to associate a packet with a specific application at each end
- Still unreliable:
  - Duplication, loss, out-of-orderness possible

UDP datagram

0	16	31
Source Port	Destination Port	
Length	Checksum	
Application data		

Field	**Purpose**
Source Port	16-bit port number identifying originating application
Destination Port	16-bit port number identifying destination application
Length	Length of UDP datagram (UDP header + data)
Checksum	Checksum of IP pseudo header, UDP header, and data

Typical applications of UDP

- Where packet loss etc is better handled by the application than the network stack
- Where the overhead of setting up a connection isn't wanted

- VOIP
- NFS – Network File System
- Most games

TCP

- Reliable, *full-duplex*, *connection-oriented*, *stream* delivery
  - Interface presented to the application doesn't require data in individual packets
  - Data is guaranteed to arrive, and in the correct order without duplications
    - Or the connection will be dropped
  - Imposes significant overheads

Applications of TCP

- Most things!
  - HTTP, FTP, …

- Saves the application a lot of work, so used unless there’s a good reason not to

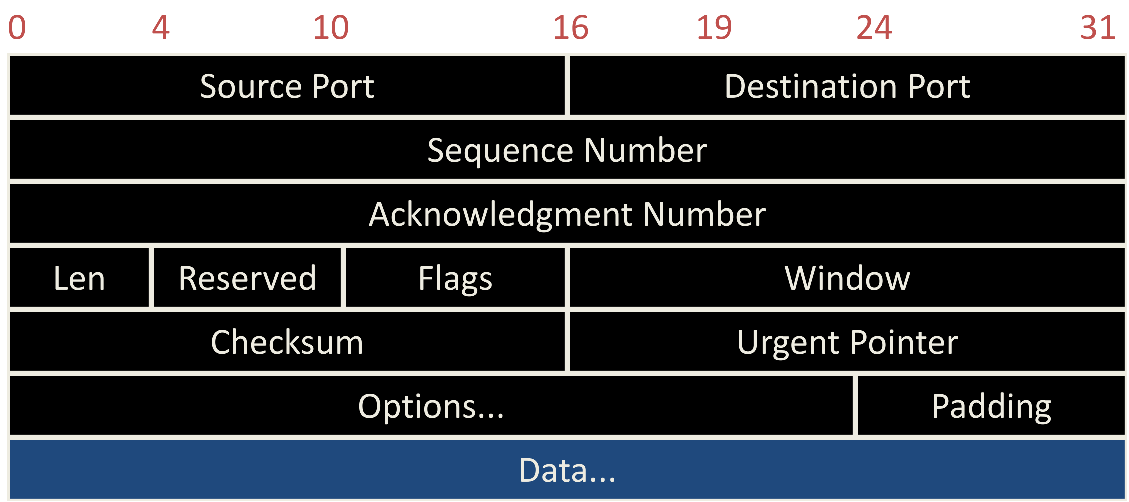
TCP implementation

- Connections are established using a *three-way handshake*
- Data is divided up into packets by the operating system
- Packets are numbered, and received packets are acknowledged
- Connections are explicitly closed
  - (or may abnormally terminate)

TCP Packets

- Source + destination ports
- Sequence number (used to order packets)
- Acknowledgement number (used to verify packets are received)

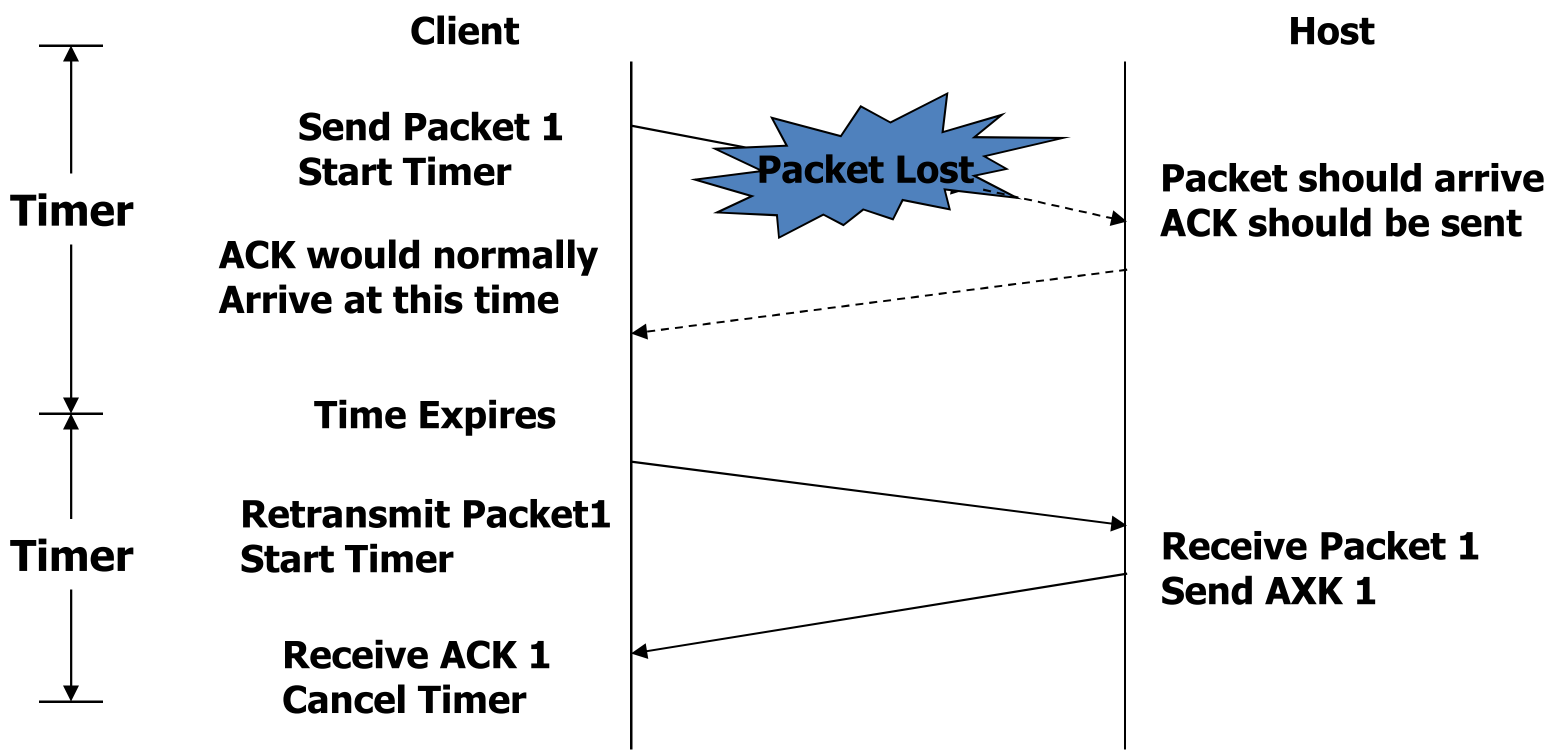
TCP Segment



Field	**Purpose**
Source Port	Identifies originating application
Destination Port	Identifies destination application
Sequence Number	Sequence number of first octet in the segment
Acknowledgment #	Sequence number of the next expected octet (if ACK flag set)
Len	Length of TCP header in 4 octet units
Flags	TCP flags: SYN, FIN, RST, PSH, ACK, URG
Window	Number of octets from ACK that sender will accept
Checksum	Checksum of IP pseudo-header + TCP header + data
Urgent Pointer	Pointer to end of “urgent data”
Options	Special TCP options such as MSS and Window Scale

You just need to know port numbers, seq and ack are added

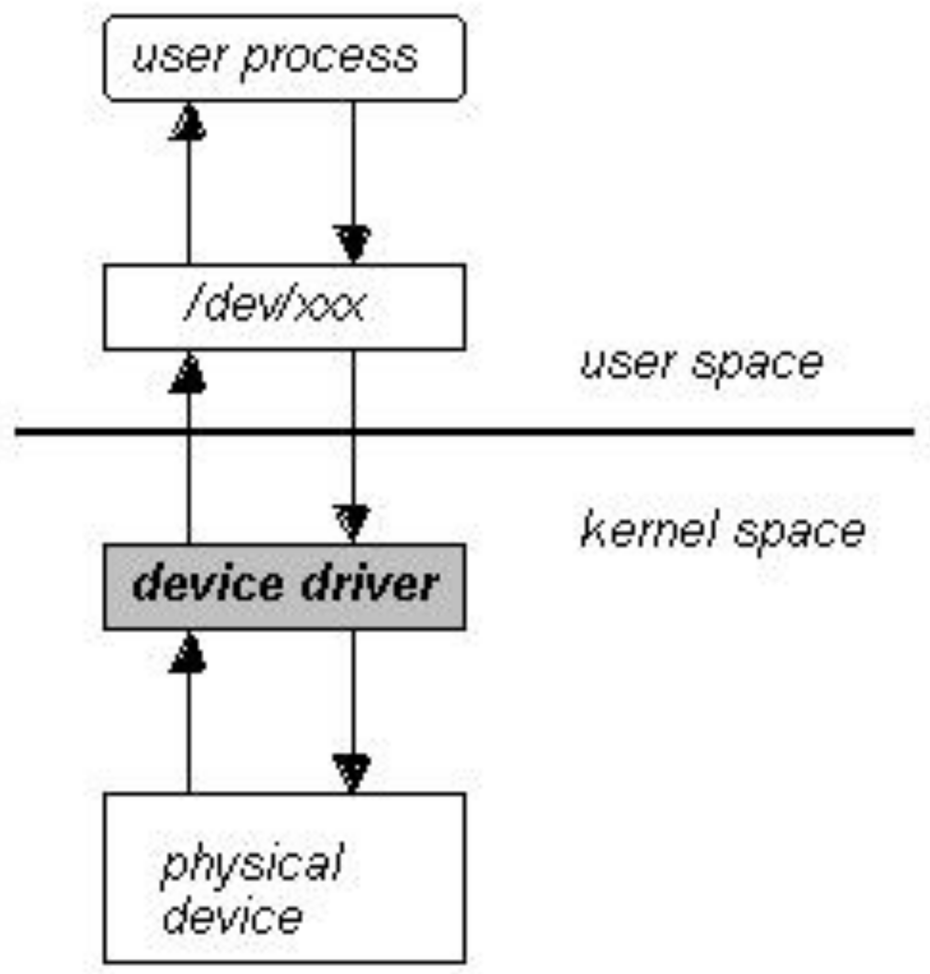
TCP : Data transfer



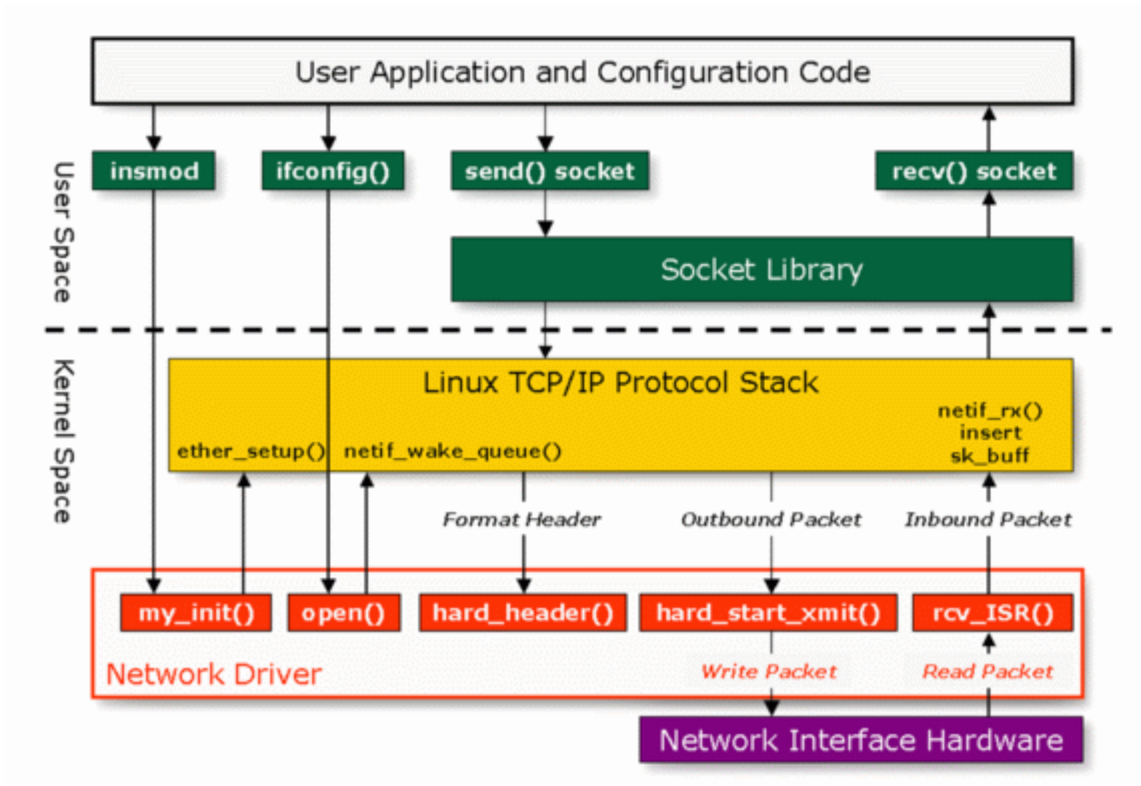
IPv6

- 128 bit addresses
  - Make it feasible to be very wasteful with address allocations
- Lots of other new features
  - Built-in autoconfiguration, security options, …
- Not really in production use yet

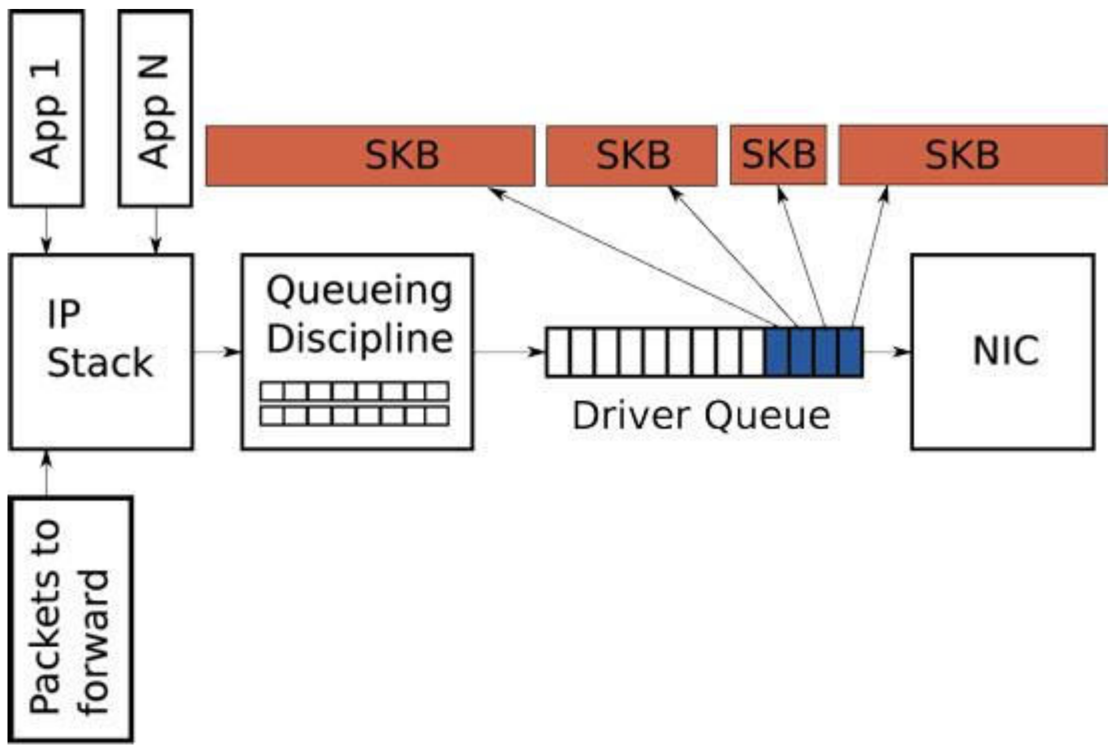
General Structure



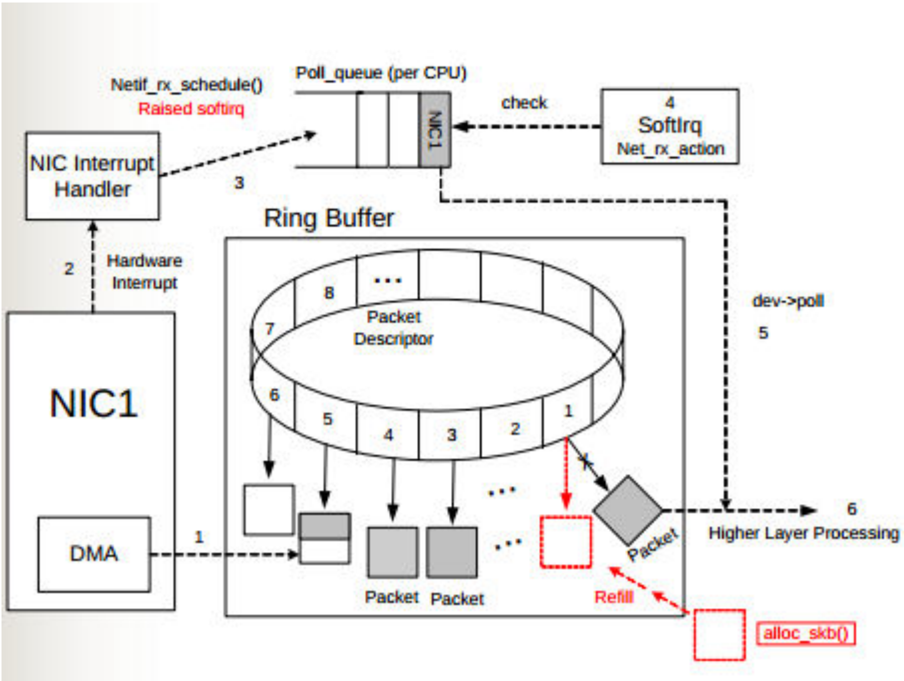
Anatomy of Network Stack



Interaction IP Stack / NIC

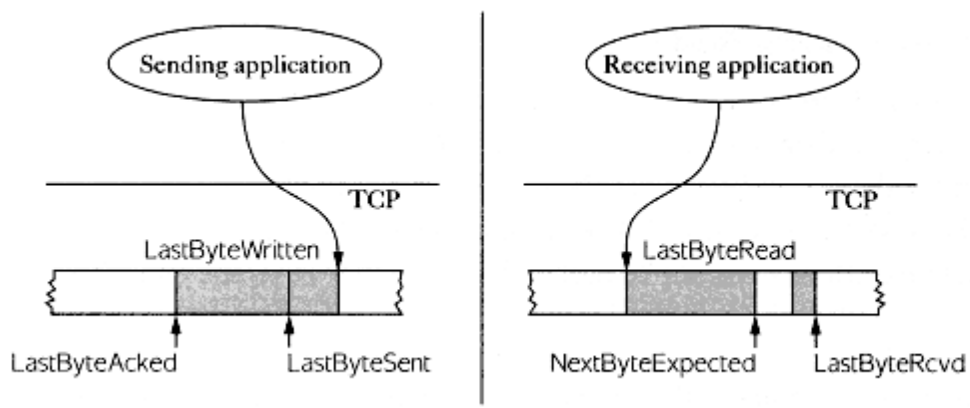


Linux Tx/Rx Ring handling



TCP/IP Details

- The sliding window serves several purposes:
- (1) it guarantees the reliable delivery of data
- (2) it ensures that the data is delivered in order,
- (3) it enforces flow control between the sender and the receiver.



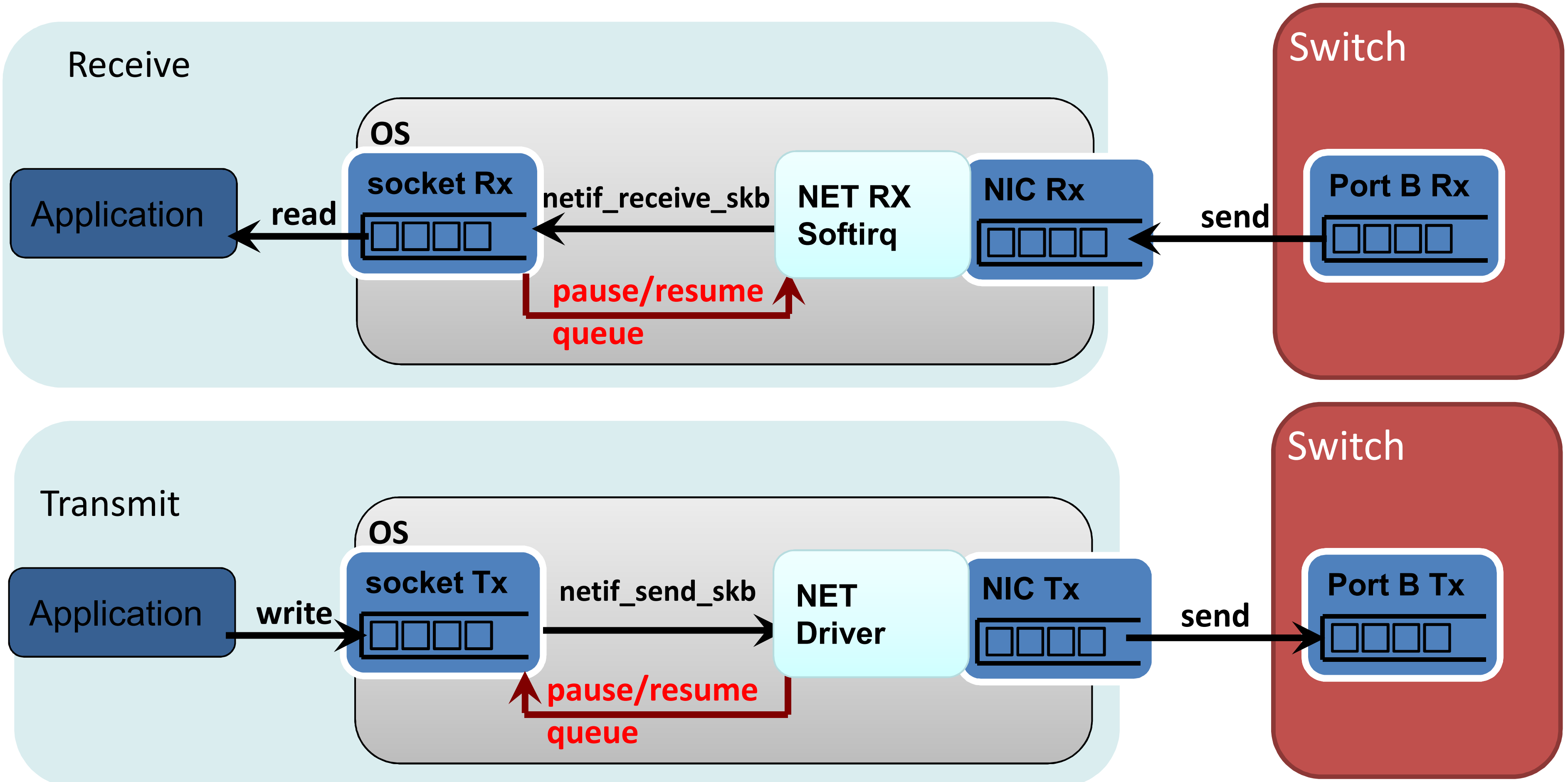
Flow Control

- Max Send and Receive Buffer sizes
- In order delivery to the consumer
- Acknowledgement of reception
- Retransmit when ack is not received in RTT (RoundTripTime) setting

Congestion Control

- Slow Start
  - Start with 1 congestion window and then doubling it
- Fast Retransmit
  - When out of order packet is received immediately ACK
- Fast Recovery

Device Driver Details

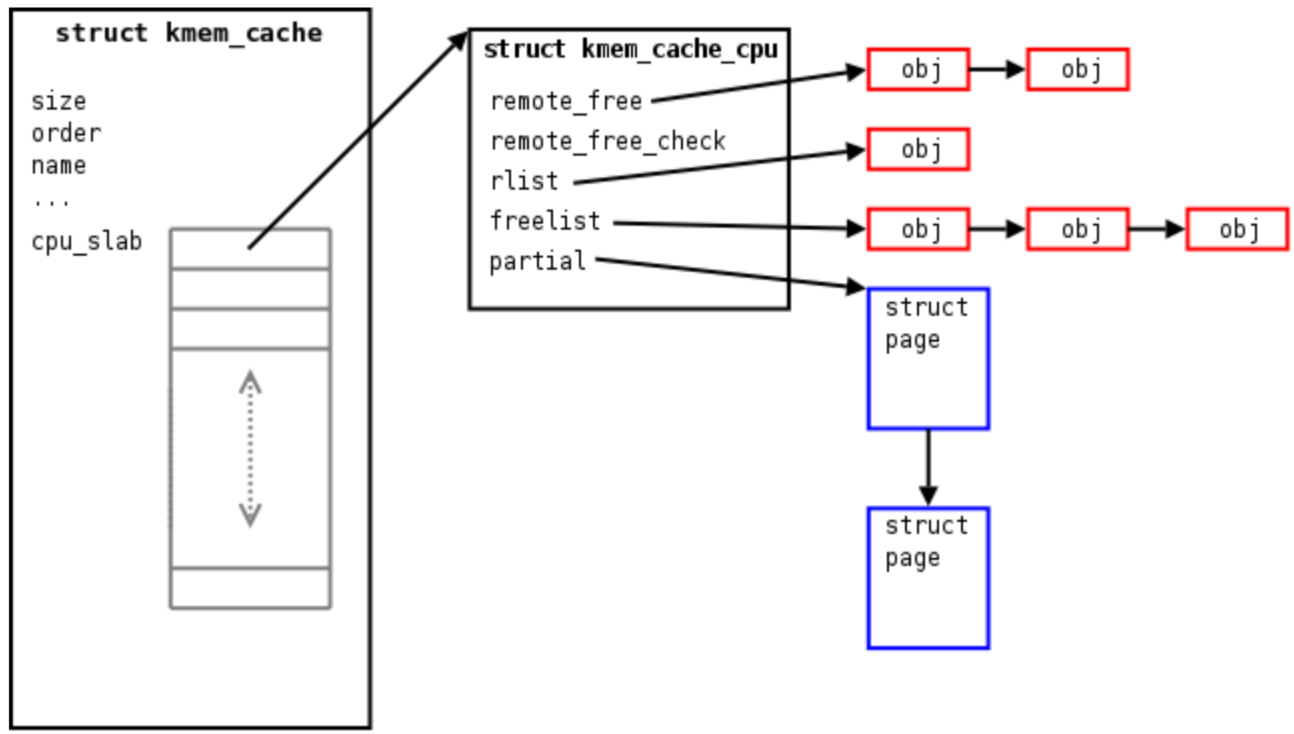


Some useful general "stuff"

- Slab-Cache
  - The primary motivation for slab allocation:
    - initialization and destruction of kernel data objects can actually outweigh the cost of allocating memory for them.
    - As object creation and deletion are widely employed by the kernel, mitigating overhead costs of initialization can result in significant performance gains.
    - "object caching" was therefore introduced in order to avoid the invocation of functions used to initialize object state.
  - Group same dynamically allocated objects under one "allocator" object
  - E.g. skb_buff_alloc()

Some useful general "stuff"

- General implementation



- Other features:
  - Coloring → Cache utilization

Linux Example of slab caches

- #> slabtop

frankeh@frankeh-vb1: ~

File Edit View Terminal Help

```
Active / Total Objects (% used)    : 194584 / 196400 (99.1%)
Active / Total Slabs (% used)      : 6944 / 6944 (100.0%)
Active / Total Caches (% used)     : 66 / 83 (79.5%)
Active / Total Size (% used)       : 40252.62K / 40617.09K (99.1%)
Minimum / Average / Maximum Object : 0.01K / 0.21K / 8.00K

  OBJS ACTIVE  USE OBJ SIZE  SLABS OBJ/SLAB CACHE SIZE NAME
 48720  48701  99%    0.13K   1624       30      6496K dentry
 41408  41407  99%    0.06K    647       64      2588K buffer_head
 33969  33968  99%    0.61K   2613       13     20904K ext4_inode_cache
 13754  13608  98%    0.09K    299       46      1196K vm_area_struct
 10240  10092  98%    0.01K     20      512        80K kmalloc-8
  9180   9176  99%    0.05K    108       85       432K sysfs_dir_cache
  6656   6440  96%    0.12K    208       32       832K kmalloc-128
  4608   4547  98%    0.02K     18      256        72K anon_vma
  3712   3458  93%    0.03K     29      128       116K kmalloc-32
  3584   3103  86%    0.02K     14      256        56K kmalloc-16
  3471   3415  98%    0.29K    267       13      1068K radix_tree_node
  2893   2889  99%    0.35K    263       11      1052K inode_cache
  2240   2158  96%    0.06K     35       64       140K kmalloc-64
  1580   1570  99%    0.38K    158       10       632K proc_inode_cache
  1530   1517  99%    0.04K     15      102        60K Acpi-Operand
   968    937  96%    0.50K    121        8       484K kmalloc-512
   966    958  99%    0.09K     23       42        92K kmalloc-96
   861    791  91%    0.19K     41       21       164K kmalloc-192
   782    778  99%    0.45K     46       17       368K shmem_inode_cache
   666    653  98%    0.44K     74        9       296K mm_struct
   624    621  99%    2.00K     78        8      1248K kmalloc-2048
```