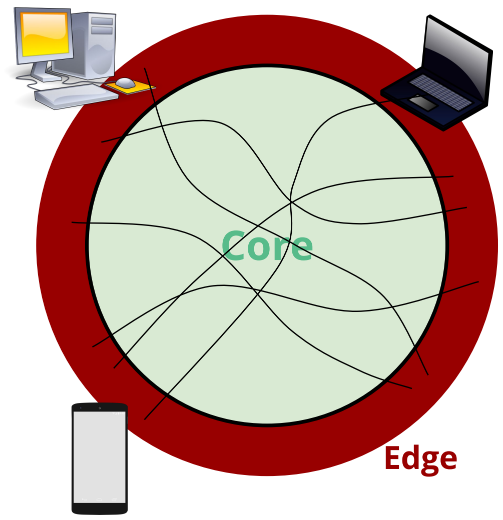
Chapter 04
Network Layer

Data Plane

Internet Layers

Application	*message exchange between two apps*
Transport	*data transfer between two processes*
Network	***data transfer between two hosts.*** ***Each host is identified by its unique IP address***
Link	*data transfer between two neighboring elements*
Physical	*bit transfer via physical medium*

Network Edge vs. Network Core



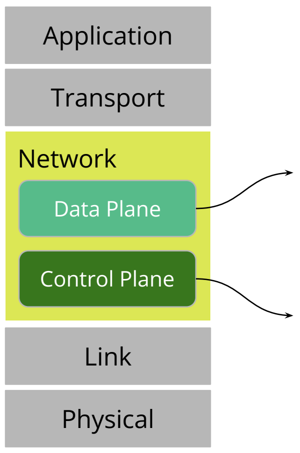
Core: interconnected routers

Edge: hosts (computing nodes) connected to the network core

Network Core vs. Network Edge

- Both the Application and Transport Layers deal with the Network Edge
  - Transport Layer: delivery of data from **process to process**
- The Network Layer deals mostly with the Network Core
  - Delivery of data from **host to host**
  - **How do you find the path from one host to another?**
- The job of navigation is carried out **collectively** by **the routers**
  - **There is no ONE centralized algorithm that controls all the routers**
  - **The navigation work is distributed across millions of routers**

Network Layer: Data and Control Planes



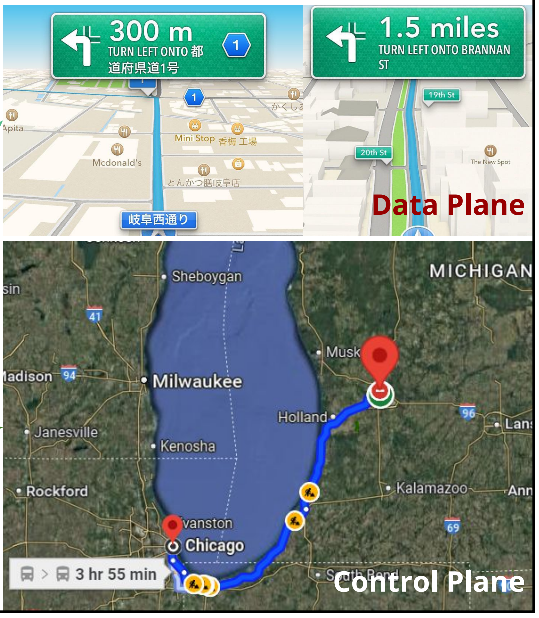
*Ch 4: **IP protocol**, ICMP protocol address format, packet handling*

Ch 5: Routing Algorithm

Network Layer

Computer Network	Road Network
Host	City / Place
Data Links	Roads/Highways
Routers	Intersections/Interchanges
Packet Forwarding*	Local Driving Decision at a specific place in your path
Packet Routing	Global Driving Direction to destination

Do get mixup with Transport Layer Demultiplexing



(Other) Network Service Model

	Guarantee of			
	Bandwidth	**No Loss**	**Order**	**Timing**
Basic Internet ("Best Effort")	❌	❌	❌	❌
Internet InterServ (RFC1633: Integrated Real-Time Services by *Xerox PARC*)	✅	✅	✅	✅
Internet DiffServ (RFC2475: Differentiated Services by *Lucent Technology*)	Possibly	Possibly	Possibly	❌

Nevertheless, the "Best Effort" model has been proven to be popular and successful!

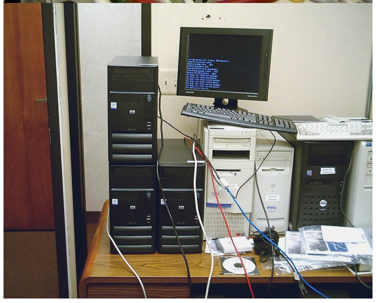
Primary Jobs of the Network Core (Recall Ch01)

- Forwarding
    - **Local action** performed by ***each individual router*** within the Network Core, moving a packet from an incoming input link to appropriate output link
    - Mapping from input to output is done via a forwarding table
- Routing
    - **Global action** (by a routing algorithm) performed ***collectively by routers*** within the Network Core, determine the path(s) taken by packets from source to destination
    - Output of a routing algorithm is used to update the individual forwarding tables of affected routers

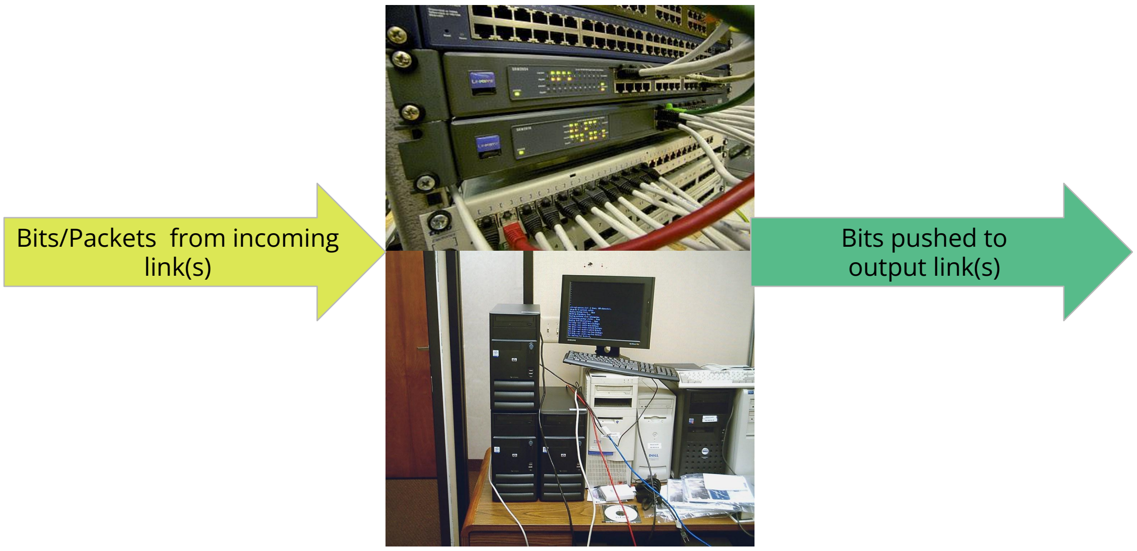
Packet Forwarding

Packet Forwarding by a Router

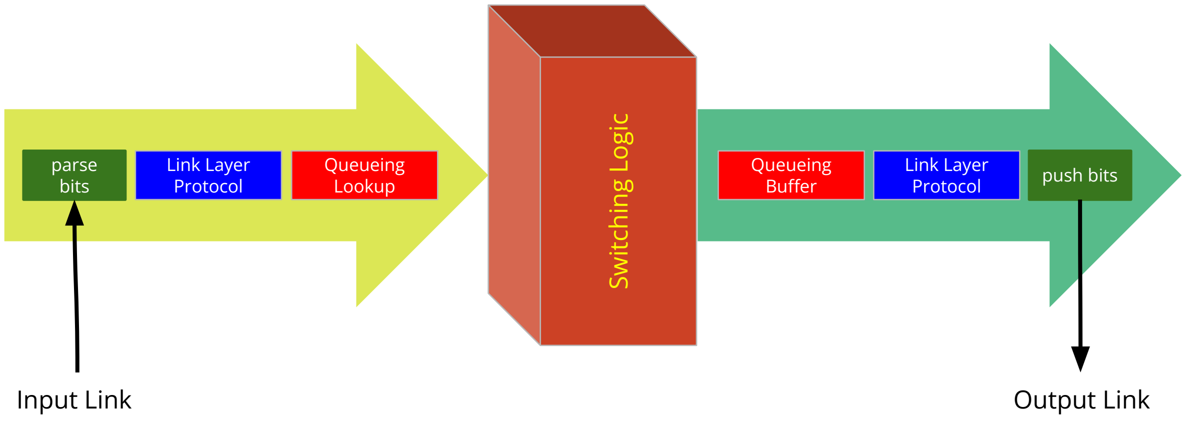
Dedicated Router Hardware


General Purpose Computer

Packet Forwarding by a Router

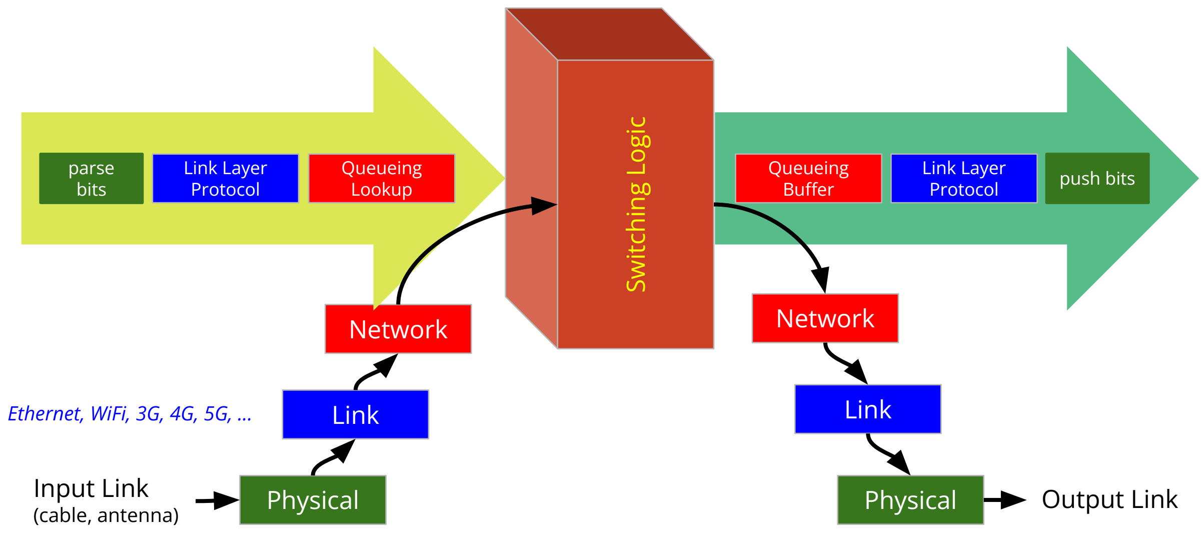

Bits/Packets from incoming link(s)
Bits pushed to output link(s)

Router Architecture

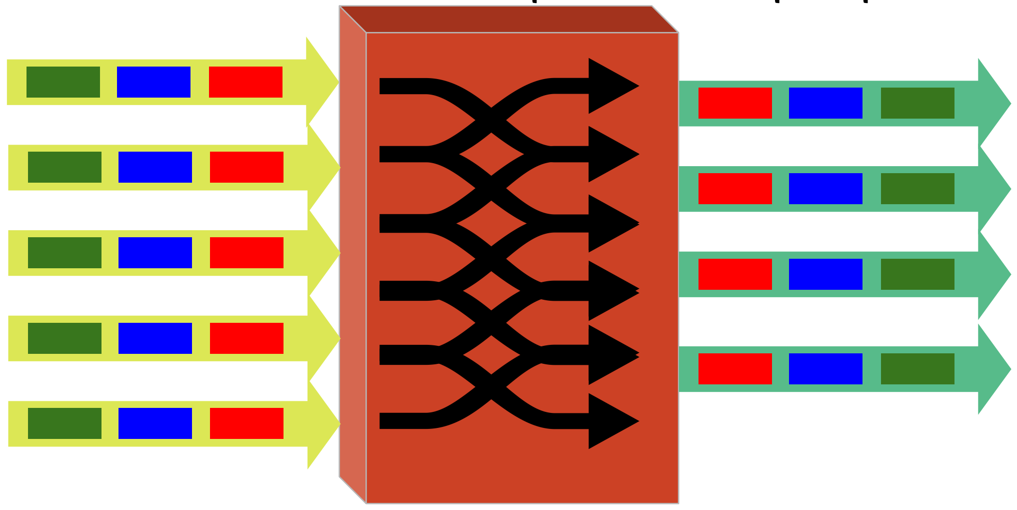


Also: Kurose & Ross Slide #14

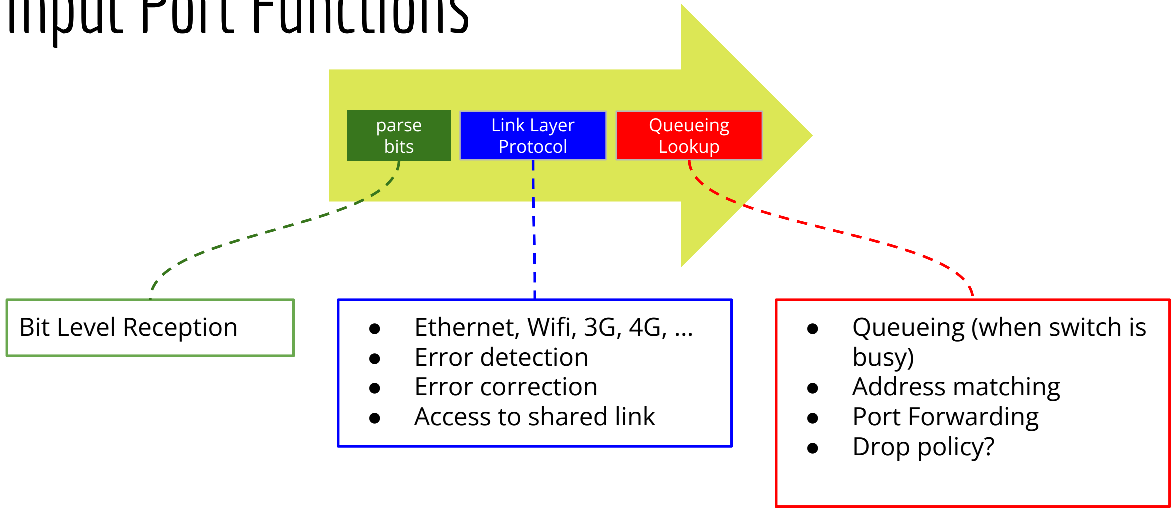
Router Architecture (1 input + 1 output port)



Router Architecture (M input + N output ports)



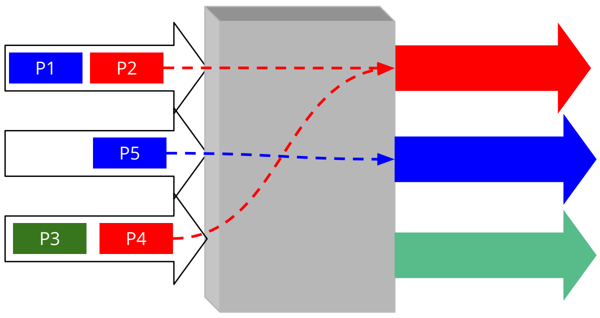
Input Port Functions



- Ethernet, Wifi, 3G, 4G, ...
- Error detection
- Error correction
- Access to shared link

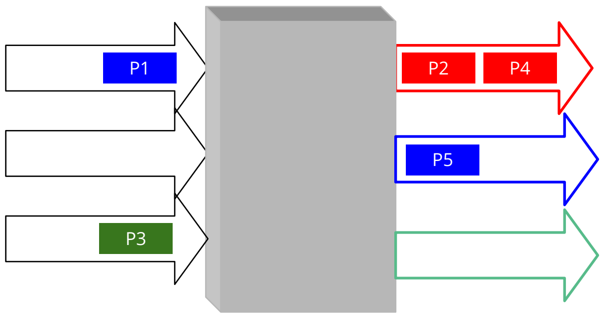
- Queueing (when switch is busy)
- Address matching
- Port Forwarding
- Drop policy?

Input Port Queuing



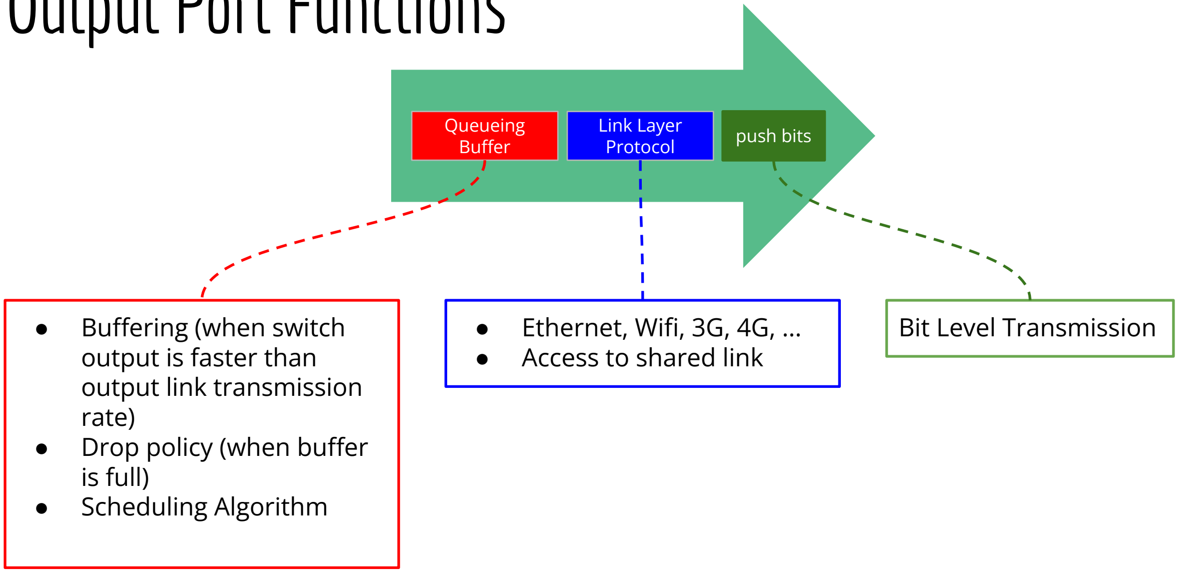
- P1 is blocked behind P2
- P3 is blocked behind P3

Output Port Queuing



- P2 and P4 must be queued at the red output port
- How to decide the order?

Output Port Functions



- Buffering (when switch output is faster than output link transmission rate)
- Drop policy (when buffer is full)
- Scheduling Algorithm

- Ethernet, Wifi, 3G, 4G, …
- Access to shared link

Bit Level Transmission

Packet Forwarding Techniques

1. **Destination Based**: Forward the packets based only on the destination IP address

2. **Generalized Forwarding**: Forward the packets using any fields in the packet header (type, size, priority, …)

Demo: Side-by-Side `traceroute`

Destination-Based Forwarding

- Use address ranges in the routing table
- Use address ranges & **longest prefix match**

See also Kurose & Ross slides page 17-29

Forwarding Table: Addr. Range vs. Addr. Prefix

Use 16-bit address in the following table (IPv4 uses 32-bit address, IPv6: 128 bits)

Min Address	Max Address	Output Port
1000 1100 0100 0000	1000 1100 0111 1111	2
1101 1010 0000 0000	1101 1011 1111 1111	0
1101 1010 1001 0000	1101 1011 1001 1111	3

$\xrightarrow{\text{simplify}}$

Address Prefix	Output Port
1000 1100 01xx xxxx	2
1101 101x xxxx xxxx	0
1101 1010 1001 xxxx	3

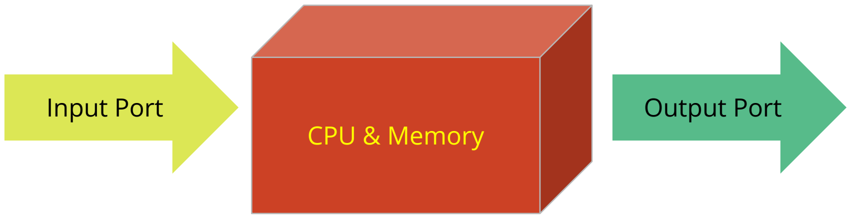
US ZIP Code (+4): Longest Prefix Match

49401	⇒	Allendale
49401-8000	⇒	Campus View Apartments
49401-9403	⇒	Grand Valley State University
49401-80xx	⇒	Further west of GVSU

Switching Hardware

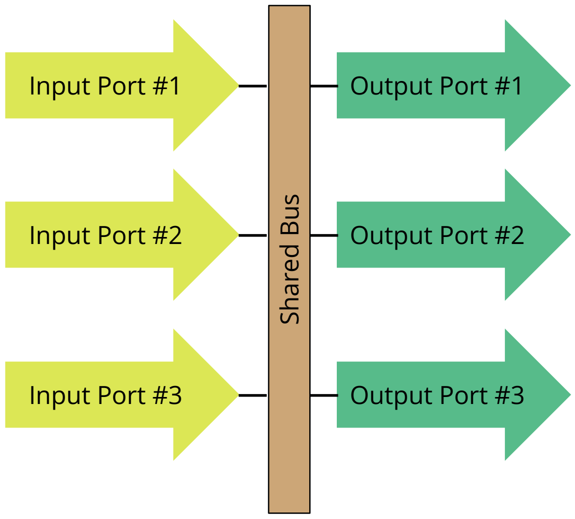
- How to connect input ports to output ports?
- If a router has N input ports and N output ports. How many of them ports can be *used in parallel*?
    - Only one input-output pair at a time
    - Only some of the N input-output pairs at a time
    - All N input-output pairs at the same time
- Three switching techniques
    - Memory (first generation routers)
    - Bus
    - Interconnection network, crossbar switches
        - Single-stage
        - Multi-stage (CLOS network)

Switching via Memory



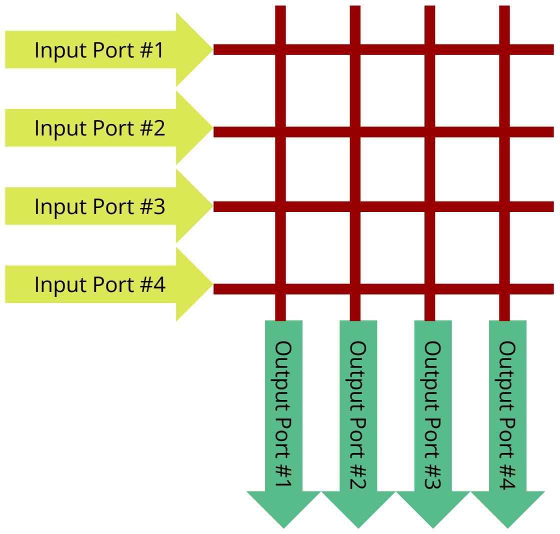
- A traditional CPU performs the switching logic
- Copy packet from input port to memory
- Determine output port
- Copy packet from memory to output port
- **Switching speed is at most 50% memory bandwidth**
    - 2 copy operations per packet

Switching via Shared Bus



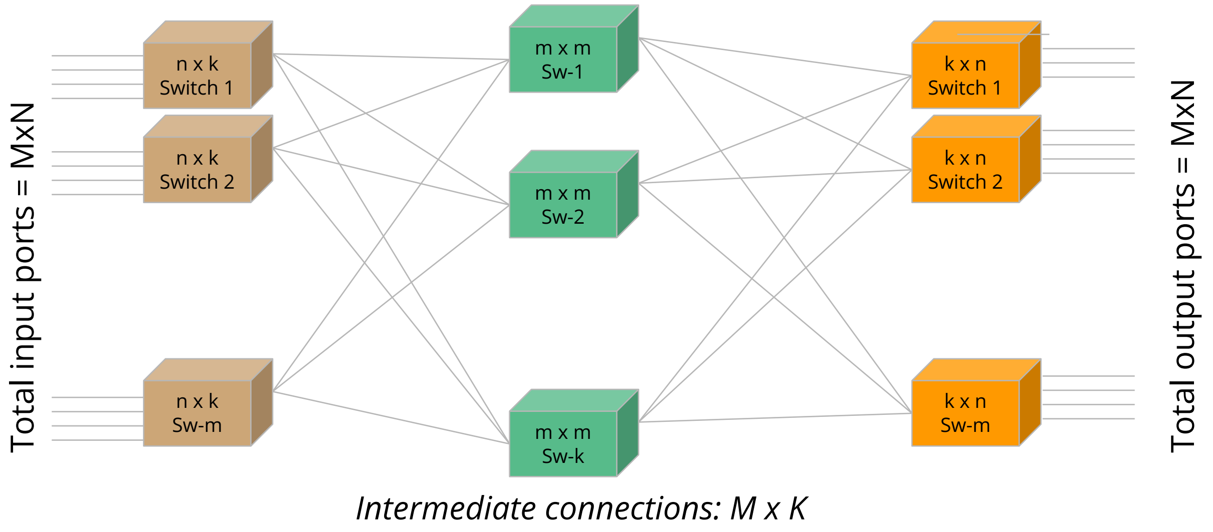
- Packets flow from input port to output port via a shared bus
- No copying of packets needed
- Only one pair of (input, output) port can be active at anytime
- **Switching speed is limited by bus bandwidth**

Switching via Crossbar Switches



- Packets flow from input port to output port via crossbar switches
- Several pairs of (input, output) port can be active if they are independent
- Require N^2 switches to "connect" one input port to a destination output port
  - Solution: use multiple stages of smaller switches

CLOS Network (Charles Clos, 1952)



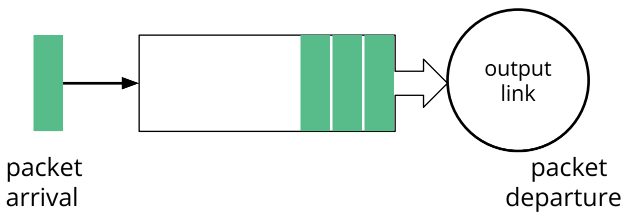
Intermediate connections: M x K

Output Buffer Management

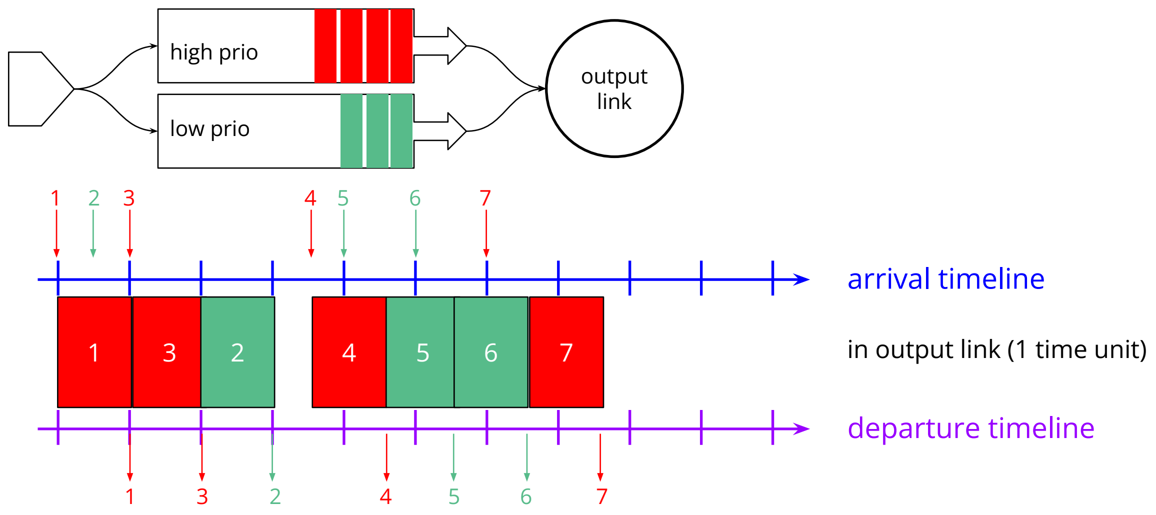
- Drop/Add Policy: which packet to add/drop when buffer is full
  - Tail drop: drop arriving packet
  - Priority: drop based on “some criteria” (age, size, type, ...)
- Scheduling Policy: which packet to transmit (push) onto the link
  - FCFS
  - **Priority:** Based on what fields/packet properties?
  - Round Robin
  - Weighted Fair Queueing (Generalized Round-Robin)

Also: Kurose & Ross Slides 35-38

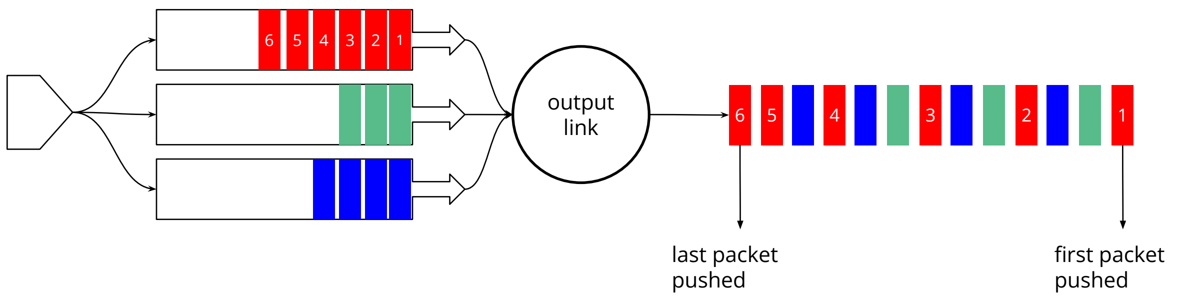
Packet Scheduling Model



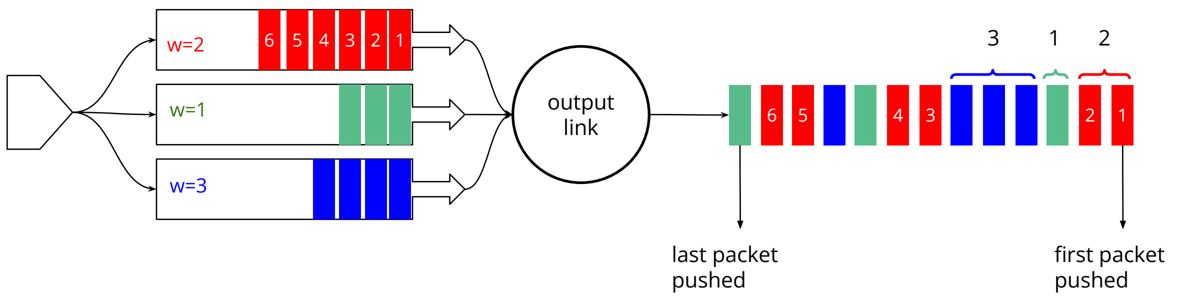
Packet Scheduling: Priority



Packet Scheduling: Round-Robin



Packet Scheduling: Weighted Fair Queueing

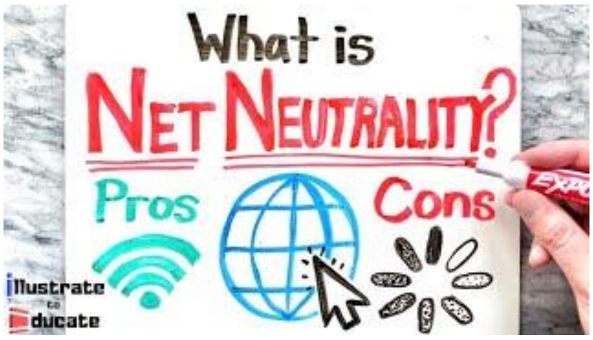


Possible Causes of Packet Drop

- Received bits on incoming link are corrupted
- **Buffer**/Queue on the incoming link is **full**
- Failed lookup of destination host IP address
- **Buffer**/Queue on the outgoing link is **full**
- Transmitted bits on the outgoing are corrupted
- *Time to Live becomes zero (specific to IP protocol)*

How big should be the buffer to minimize dropped packets?

Net Neutrality



Buffer Size Contributing Factors

- **Link capacity (C)**: Higher link capacity ⇒ More frequent packet arrival ⇒ Require more buffer space
- **Round-Trip Time (RTT)**: Packets may have to be retained until ACK is received ⇒ Longer RTT ⇒ Require more buffer space
- **Number of Sender-Receiver Flows (N)**: higher number of sender-receiver flows tends to drain the output buffer more frequently ⇒ More flow ⇒ Less buffer space needed

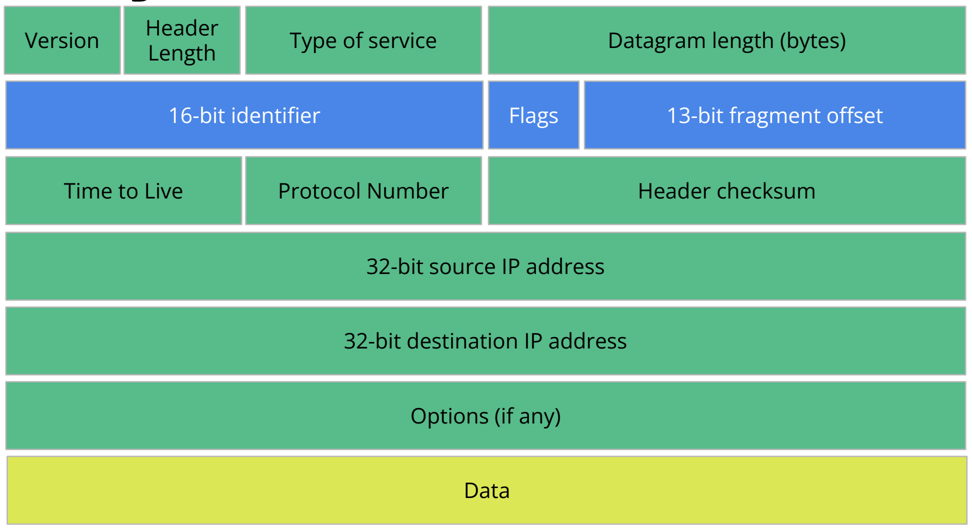
$$\text{Buffer Size} = \frac{RTT \cdot C}{\sqrt{N}}$$

The Internet Protocol (IPv4, IPv6)

Relevant RFCs

- RFC791 (Internet Protocol version 4)
- RFC2460, RFC4291 (IP version 6)
- RFC792 (Internet Control Message Protocol)

IPv4 Datagram Format



Live Demo: IP Addresses (Ranges)

```
nslookup gvsu.edu
nslookup mail.gvsu.edu
nslookup lms.gvsu.edu

# On MacOS
ipconfig getiflist
ipconfig getifaddr en___
ipconfig getsummary en0
```

```
# Online (From Web Browser)
whois 148.61.0.0
whois 104.17.0.0
whois 142.250.190.142
```

IPv4 Address: binary & dot notation

`gvsu.edu` `104.17.87.18`

$$\underbrace{\mathbf{0110\ 1000}}_{\mathbf{104}}\quad\underbrace{\mathbf{0001\ 0001}}_{\mathbf{17}}\quad\underbrace{\mathbf{0101\ 0111}}_{\mathbf{87}}\quad\underbrace{\mathbf{0001\ 0010}}_{\mathbf{18}}$$

`part of CloudFlare Inc subnet`
`104.16.0.0 - 104.31.255.255`

Organizing Phone Numbers/Room Numbers

+16163312919

+1-616-3312919

+1-616-331-2919 **+1-616-331**-3255

Common prefix

MAKD1117

MAK D-1-117 **MAK D-1**-189

GVSU.edu (hosted @ Cloudflare Inc)

Cloudflare subnet **104.16.0.0/12**

$$\overbrace{\underbrace{0110\ 1000}_{104}\ \underbrace{0001\ 0000}_{16}}^{\textit{12 bits}}\ \underbrace{0000\ 0000}_{0}\ \underbrace{0000\ 0000}_{0}$$

Lowest Address

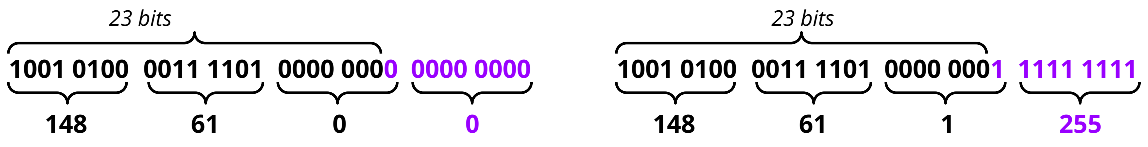
$$\overbrace{\underbrace{0110\ 1000}_{104}\ \underbrace{0001\ 1111}_{31}}^{\textit{12 bits}}\ \underbrace{1111\ 1111}_{255}\ \underbrace{1111\ 1111}_{255}$$

Highest Address

Net Range: 104.16.0.0 – 104.31.255.255

GVSU.edu (Under Merit Network)

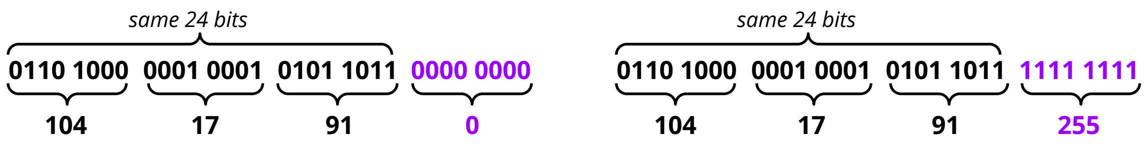
whois 148.61.0.0 ⇒ Merit subnet **148.61.0.0/23**



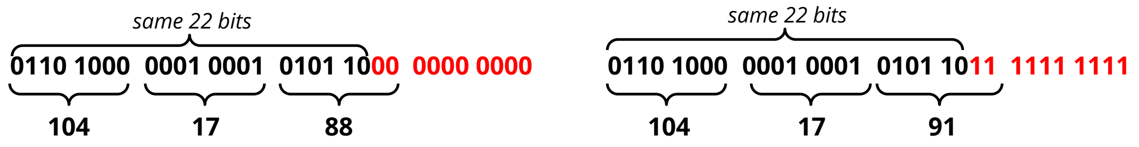
Net Range: 148.61.0.0 – 148.61.1.255

IPv4 Address Range (CIDR Notations)

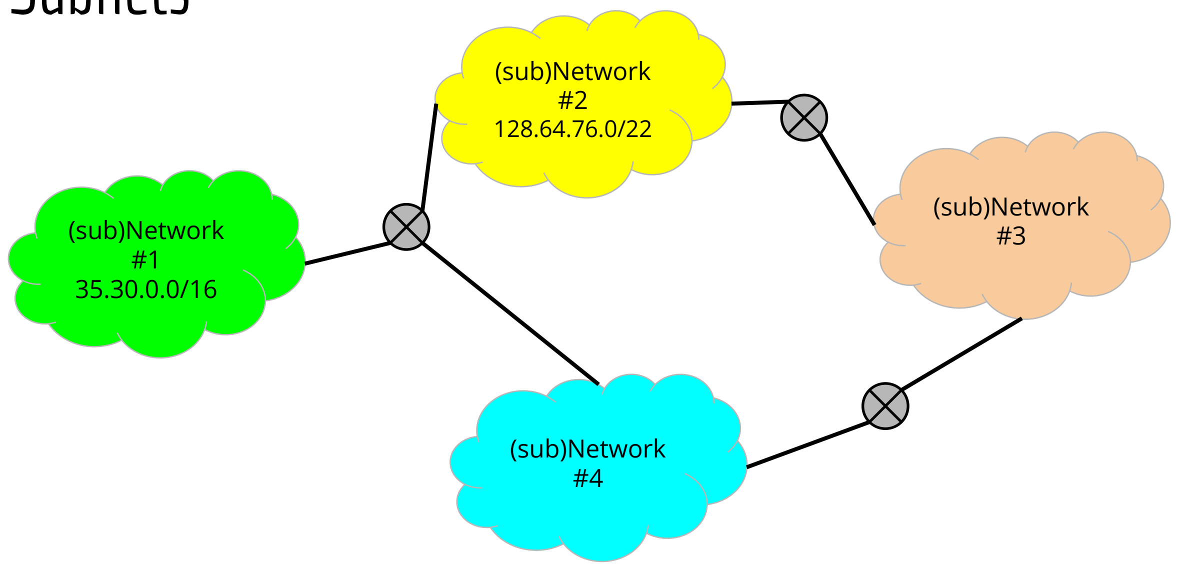
Subnet **104.17.91/24**



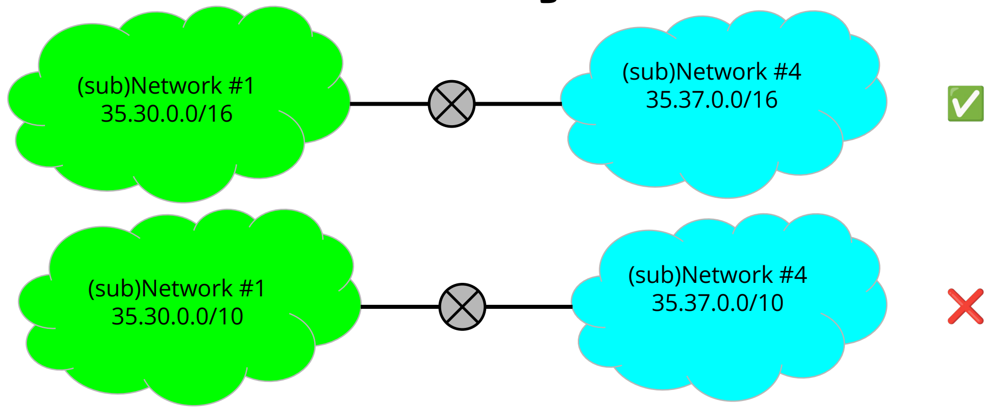
Subnet **104.17.88/22**



Subnets

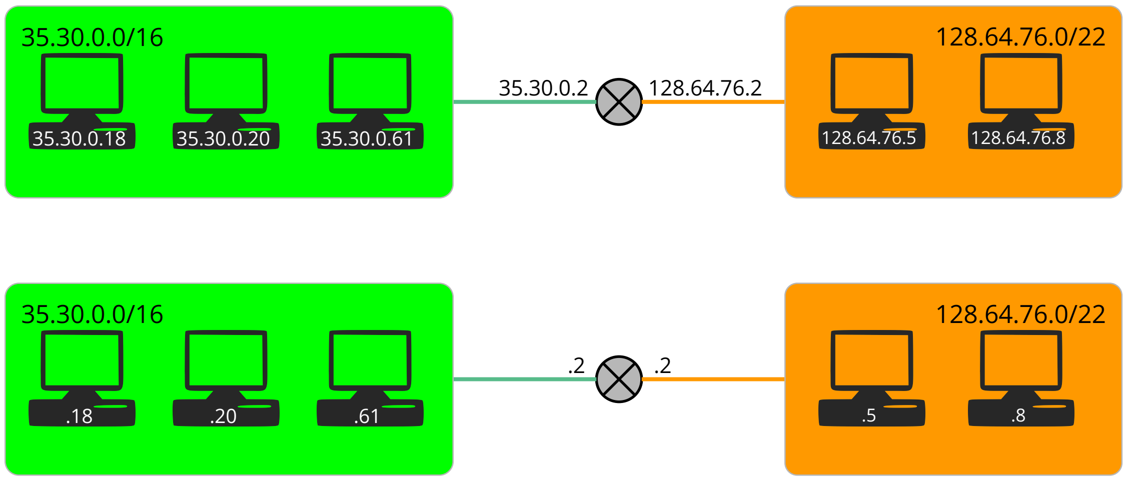


Subnets: Good vs. Poor Designs


(sub)Network #1 35.30.0.0/16
(sub)Network #4 35.37.0.0/16
✅
(sub)Network #1 35.30.0.0/10
(sub)Network #4 35.37.0.0/10
❌

```
35.30.0.0 ⇒ 0010 0011   0001 1110   0000 0000   0000 0000
35.37.0.0 ⇒ 0010 0011   0010 0101   0000 0000   0000 0000
```

Subnets: Shortened Address Notation



Broadcast Addresses (Highest Addr in Subnet)

	IP Address	Forwarded to other subnets
Local Broadcast	255.255.255.255	No
Directed Broadcast	Last/Highest address in a subnet	Yes

Subnet	Address (Binary)	Directed Broadcast Address
200.14.32.0/20	**L: 1100 1000 0000 1110 0010 0000 0000 0000** **H: 1100 1000 0000 1110 0010 1111 1111 1111**	200.14.47.255
200.14.32.0/24	**L: 1100 1000 0000 1110 0010 0000 0000 0000** **H: 1100 1000 0000 1110 0010 0000 1111 1111**	200.14.32.255
200.14.32.0/26	**L: 1100 1000 0000 1110 0010 0000 0000 0000** **H: 1100 1000 0000 1110 0010 0000 0011 1111**	200.14.32.63

Private IP Addresses

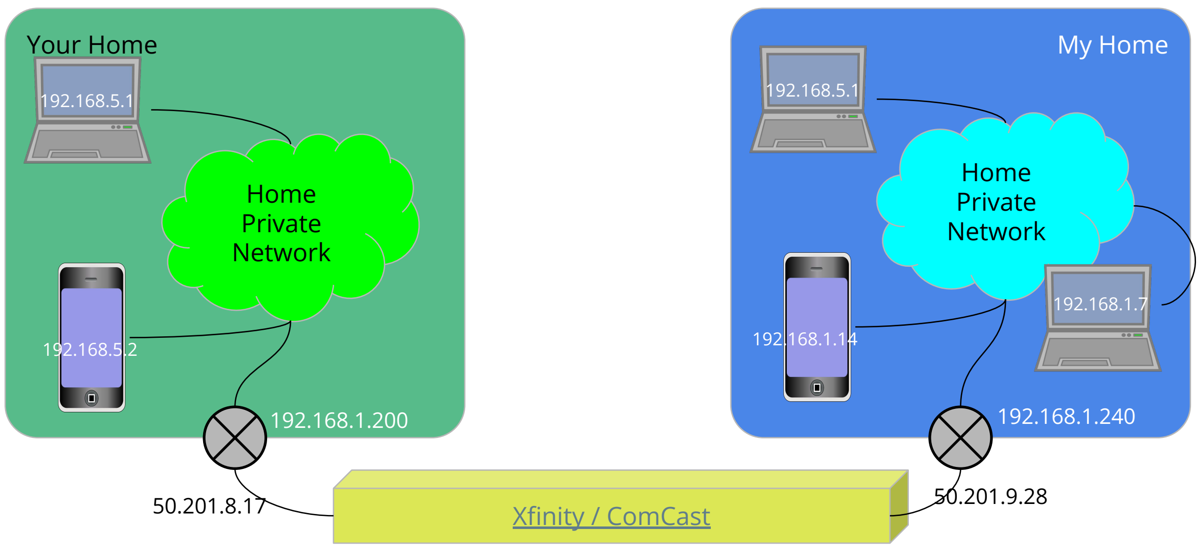
Type	Notation	Range of 32-bit addresses	#bits for host address	
Class A	10.0.0.0/8	`00001010 xxxxxxxx yyyyyyyy zzzzzzzz` First byte: 0x0A	24 = 32 - 8	2^{24} - 1 hosts
Class B	172.16.0.0/12	`10101100 0001xxxx yyyyyyyy zzzzzzzz` First byte: 0xAC, underlined is a 'B'	20 = 32 - 12	2^{20} - 1 hosts
Class C	192.168.0.0/16	`11000000 10101000 yyyyyyyy zzzzzzzz` First byte: 0xC0	16 = 32 - 16	2^{16} - 1 hosts

- Theoretical limit is 2^k - 1, because the highest address in the subnet is reserved for the broadcast address
- Practical limit is 2^k - 2, in addition to the broadcast address we must reserve one more address for the gateway within the subnet

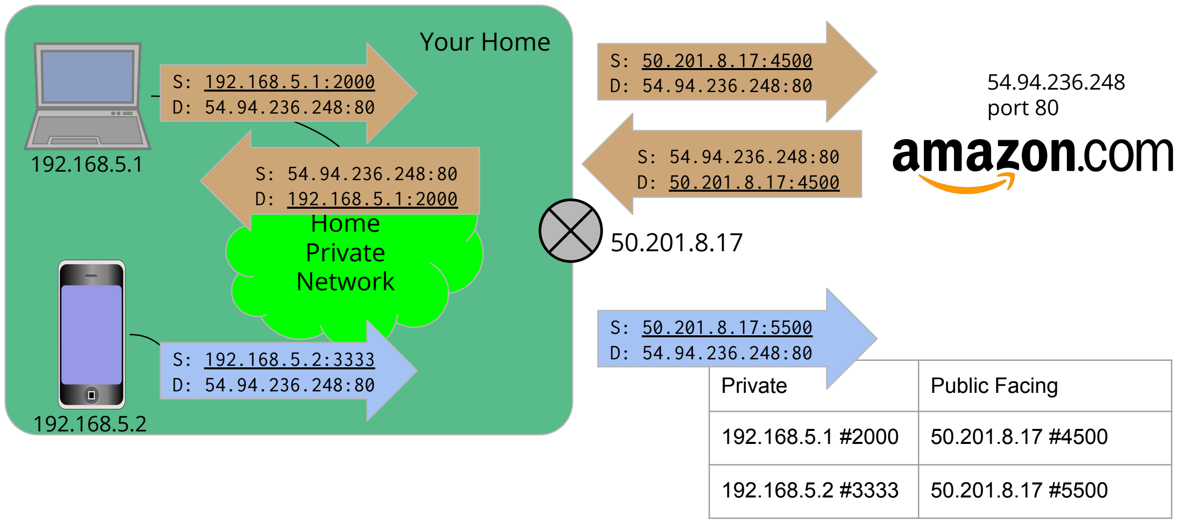
Private IP Addresses & Multicast

Type	Notation	32-bit addresses	Address Range
Class A	10.0.0.0/8	`00001010 xxxxxxxx yyyyyyyy zzzzzzzz`	10.0.0.0 - 10.255.255.255
Class B	172.16.0.0/12	`10101100 0001xxxx yyyyyyyy zzzzzzzz`	172.16.0.0 -172.31.255.255
Class C	192.168.0.0/16	`11000000 10101000 yyyyyyyy zzzzzzzz`	192.168.0.0 - 192.68.255.255
Class D	224.0.0.0/4	`1110xxxx yyyyyyyy zzzzzzzz wwwwwwww`	224.0.0.0 - 239.255.255.255

Private IP Addresses



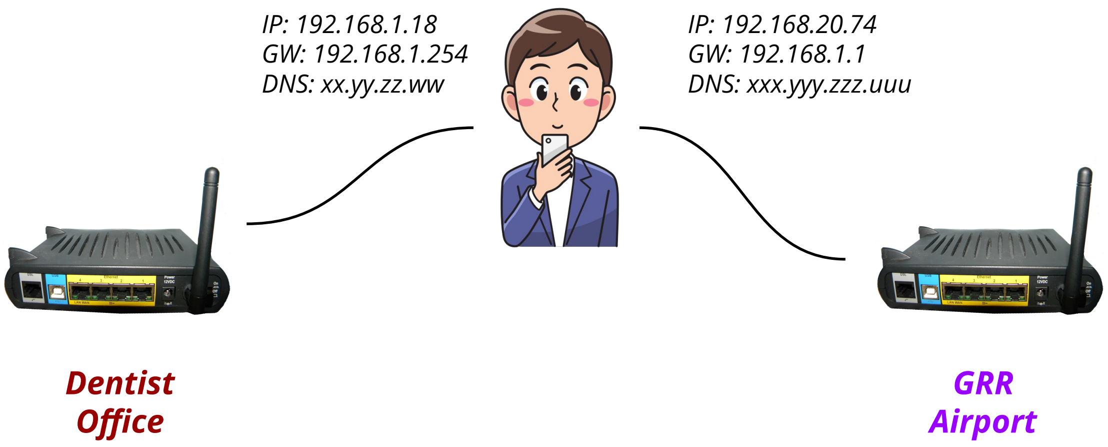
NAT: Network Address Translation (by Routers)



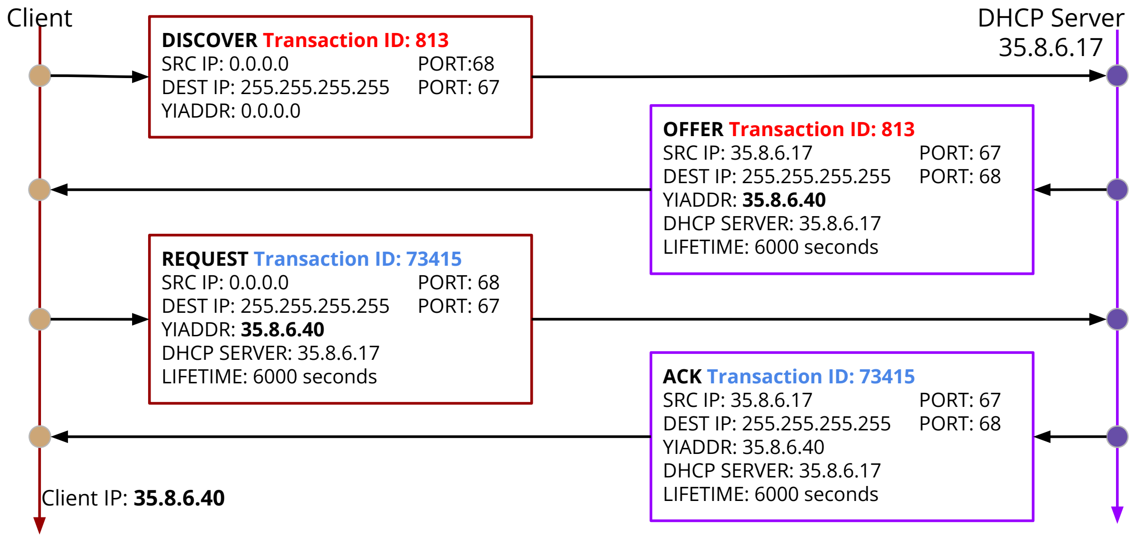
Private	Public Facing
192.168.5.1 #2000	50.201.8.17 #4500
192.168.5.2 #3333	50.201.8.17 #5500

Dynamic Host Configuration Protocol (DHCP)

Connecting Your Phone/Laptop To (Home) WiFi



DHCP Transactions (DORA)



Client
DHCP Server
35.8.6.17
DISCOVER Transaction ID: 813
SRC IP: 0.0.0.0 PORT:68
DEST IP: 255.255.255.255 PORT: 67
YIADDR: 0.0.0.0
OFFER Transaction ID: 813
SRC IP: 35.8.6.17 PORT: 67
DEST IP: 255.255.255.255 PORT: 68
YIADDR: 35.8.6.40
DHCP SERVER: 35.8.6.17
LIFETIME: 6000 seconds
REQUEST Transaction ID: 73415
SRC IP: 0.0.0.0 PORT: 68
DEST IP: 255.255.255.255 PORT: 67
YIADDR: 35.8.6.40
DHCP SERVER: 35.8.6.17
LIFETIME: 6000 seconds
ACK Transaction ID: 73415
SRC IP: 35.8.6.17 PORT: 67
DEST IP: 255.255.255.255 PORT: 68
YIADDR: 35.8.6.40
DHCP SERVER: 35.8.6.17
LIFETIME: 6000 seconds
Client IP: 35.8.6.40

ICMP
Internet Control Message Protocol

ICMP: Internet Control Message Protocol

- ICMP
- Used by host / routers for Network Layer Information
  - Errors
  - Echo request/reply (PING)
- ICMP messages are carried as payload in IP datagrams

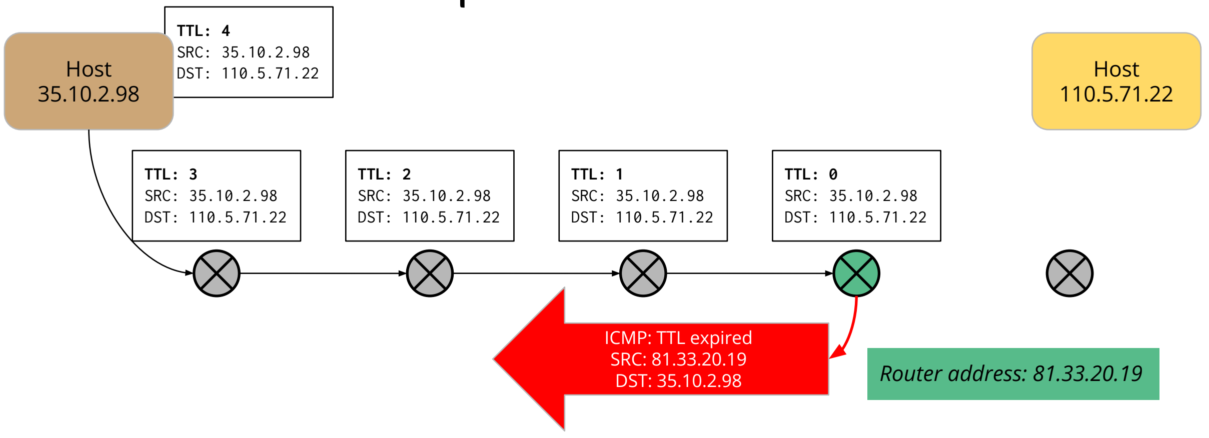
ICMP Message Type/Code

Type	Description
3	Destination Unreachable
4	No space in buffer
5	Datagram redirected
8	Echo
0	Echo reply
11	Time expired
12	Parameter error
13	Timestamp
14	Timestamp reply

Type	Code	Description
4	0	No space in buffer
5	0	Redirect datagrams for the network
	1	Protocol unreachable
	2	Port unreachable
	3	Fragmentation needed
	5	Source field failed
11	0	TTL expired
	1	Fragment reassembly time exceeded

Live Demos: ping and traceroute

IP Time-To-Live Expiration

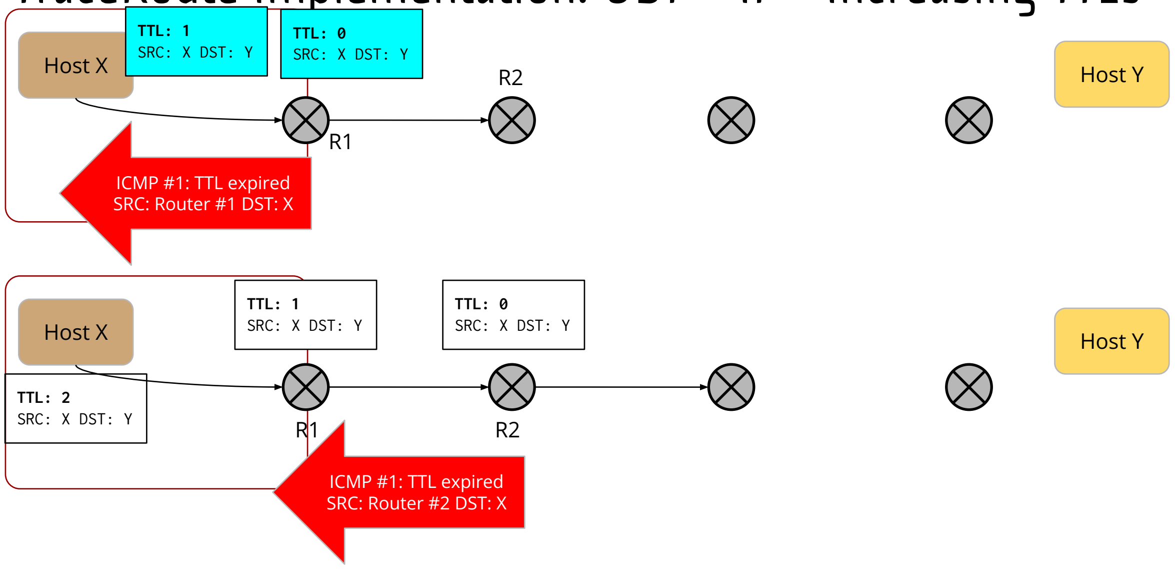


traceroute

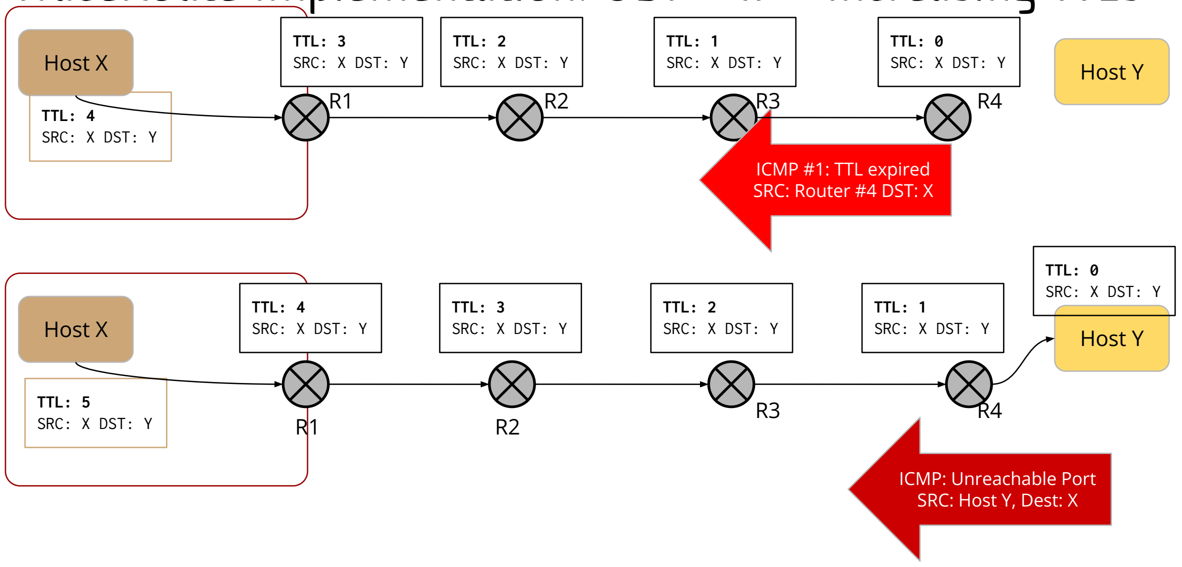
```
traceroute to computing.gvsu.edu (104.17.88.18), 30 hops max, 60 byte packets

1  * * *
2  * * *        no response within 5 seconds
3  * * *
4  * * *
5  148.61.0.126 (148.61.0.126)  2.874 ms  2.921 ms  3.661 ms
6  irbx643.gdrp-eberhard-c1.mich.net (192.122.182.137)  1.078 ms  1.078 ms  1.112 ms
7  et-5-3-0x3.dtrt-wsudc-c1.mich.net (207.72.230.47)  4.705 ms  4.739 ms  4.716 ms
8  et-4-1-5x3.sfld-cor-123net.mich.net (207.72.230.128)  4.631 ms  4.680 ms  4.612 ms
9  static.det-ix.net (209.124.52.26)  5.144 ms  5.514 ms  5.298 ms
10  104.17.88.18 (104.17.88.18)  4.581 ms  4.618 ms  4.628 ms
```

TraceRoute Implementation: UDP + IP + Increasing TTLs

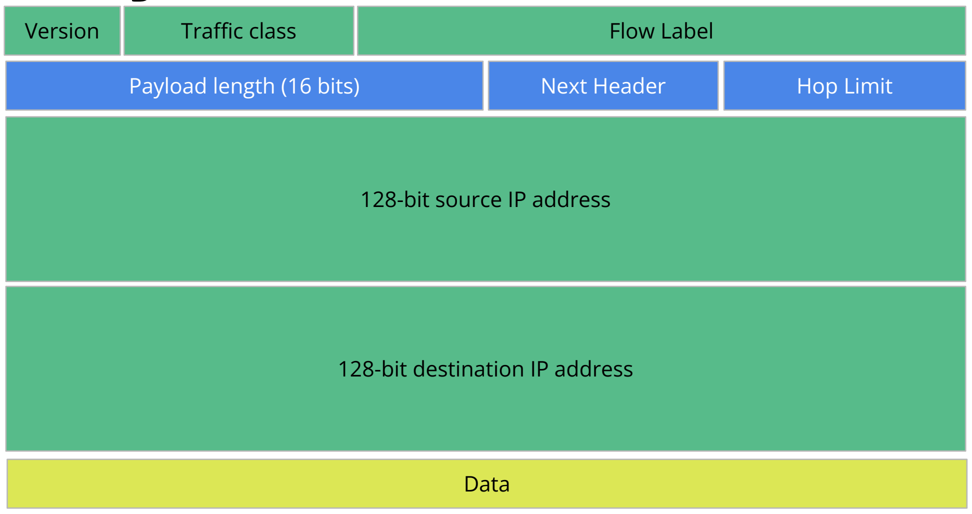


TraceRoute Implementation: UDP + IP + Increasing TTLs



IPv6

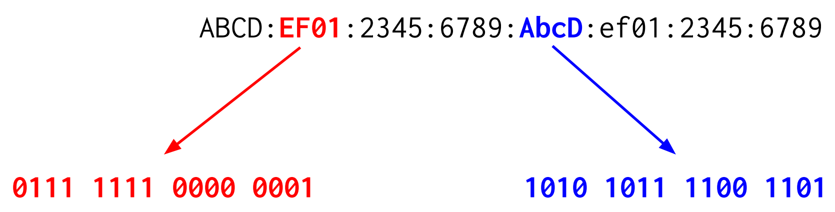
IPv6 Datagram Format



IPv4 vs IPv6

	IPv4	IPv6
Header Length	✅	❌
Header Checksum	✅	❌
Flow label	❌	✅
Type of service	✅	Traffic Class
Datagram Length	✅	Payload length
Fragmentation	✅	❌
TTL	✅	Renamed to Hop Limit
Upper Layer protocol	✅	❌
IP addresses	✅	Widen to 128 bits
Options	✅	Extension Header

IPv6 Address (128 bits ⇒ 32 hex digits)



IPv6 Address Shortening

Non-shortened notation: `ABCD:0000:0000:0004:21C7:0000:0000:6789`

Shortened	Rule Applied
`ABCD:0:0:4:21C7:0:0:6789`	Remove leading zeros
`ABCD::4:21C7:0:0:6789`	First group **zero compression**
`ABCD:0:0:4:21C7::6789`	Second group **zero compression**

Shorten only one group of zeros

$$\begin{tikzcd}
 & & \texttt{ABCD:0000:0000:04C7:0000:0000:0000:6789} \\
\texttt{ABCD::4C7::6789} & ? \arrow[ur, bend left] \arrow[dr, bend right] & \\
 & & \texttt{ABCD:0000:0000:0000:04C7:0000:0000:6789}
\end{tikzcd}$$

IPv6: Relevant RFCs

- Datagram Format
  - RFC1883 (Sep 995) ⇒ RFC2460 (Dec 1998) ⇒ RFC8200 (Jul 2017)
- Address Architecture
  - RFC4291 (Feb 2006) ⇒ RFC6052 (Oct 2010)

Invalid Shortening

Original	Incorrect Shortening	Reason
91a0:8322:0000:0000:abcd:8000:0000:61df	`91a:8322:9:0:abcd:8:0:61df`	**Trailing zeros** can't be removed
91a0:8322:0000:0000:abcd:8000:0000:61df	`91a0:8322:0:0:0:8000::61df`	Zero compression only **one 16-bit group**
91a0:0000:0000:0000:abcd:0000:0000:61df	`9a10:0::abcd:0:0:61df`	Not the the shortest possible
91a0:0000:0000:0000:abcd:0000:0000:61df	`9a10:0:0:0:abcd::61df`	Zero compression should be applied to the **longest group**
91a0:0000:0000:0028:abcd:0000:0000:61df	`91a0:0:0:28:abcd::61df`	When multiple spots of zero compression are equally possible, **use the leftmost**

IPv6 with port number

IP Address & Port 80	Explanation
`[2001:db8::1]:80`	Square brackets separate IP address from port number
`2001:db8::1.80`	Use dot or other characters
`2001:db8::1p80`	
`2001:db8::1#80`	
`2001:db8::1:80`	Zero compression implies IP address 2001:db8:0:0:0:0:0:1

Embedding IPv4 address in IPv6 address

Prefix	IPv6 address format	Suffix
32 bits	`pppp:pppp:xxxx:xxxx:00ss:ssss:ssss:ssss`	56 bits
40 bits	`pppp:pppp:ppxx:xxxx:00xx:ssss:ssss:ssss`	48 bits
48 bits	`pppp:pppp:pppp:xxxx:00xx:xxss:ssss:ssss`	40 bits
56 bits	`pppp:pppp:pppp:ppxx:00xx:xxxx:ssss:ssss`	32 bits
64 bits	`pppp:pppp:pppp:pppp:00xx:xxxx:xxss:ssss`	24 bits
96 bits	`pppp:pppp:pppp:pppp:00pp:pppp:xxxx:xxxx`	none

These are "standard" prefix lengths. The suffix bits can be used to identify individual hosts in a subnet

IPv6 Special Prefixes

Prefix	Purpose
`::1/128`	Loopback address (send to self), equivalent to 127.0.0.1 in IPv4
`::ffff:0:0/96`	IPv4-mapped addresses
`64:FF9B::/96`	NAT64 prefix for public addresses
`64:FF9B:1:/48`	NAT64 prefix for private addresses
`FC00::/7`	Local Unicast (Routable within a LAN, similar to Private IP addresses in IPv4)
`FE80::/10`	Link-Local Unicast (not Routable)
`FEC0::/10`	Site-Local Unicast (deprecated)
`FF00::/8`	Multicast (RFC4291)

IPv4 address embedding example (RFC6052)

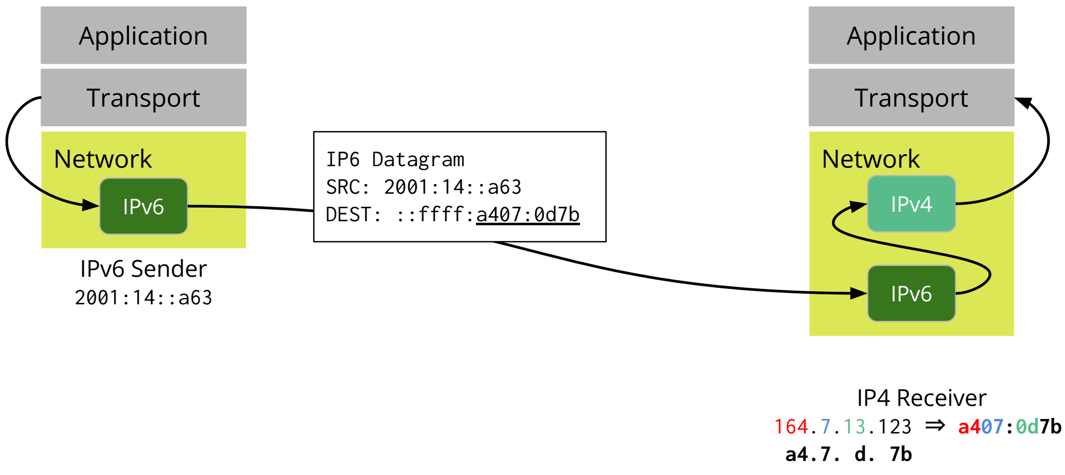
IPv4: `192.0.2.33` (in hex `C0.00.02.21` ⇒ `C000:0221`)

Network Prefix	IPv6 address format	Shortened
`2001:db8::/32`	`2001:0db8:C000:0221:0000:0000:0000:0000`	`2001:db8:c000:221::`
`2001:db8:a00::/40`	`2001:0db8:0aC0:0002:0021:0000:0000:0000`	`2001:db8:ac0:2:21::`
`2001:db8:a22::/48`	`2001:0db8:0a22:C000:0002:2100:0000:0000`	`2001:db8:a22:c000:2:2100::`
`2001:db8:a22:b00::/56`	`2001:0db8:0a22:0bC0:0000:0221:0000:0000`	`2001:db8:a22:bc0:0:221::`
`2001:db8:a22:b44::/64`	`2001:0db8:0a22:0b44:00C0:0002:2100:0000`	`2001:db8:a22:b44:c0:2:2100::`
`2001:db8:a22:b44::/96`	`2001:0db8:0a22:0b44:0000:0000:C000:0221`	`2001:db8:a22:b44::c000:221` `2001:db8:a22:b44::192.0.2.33`

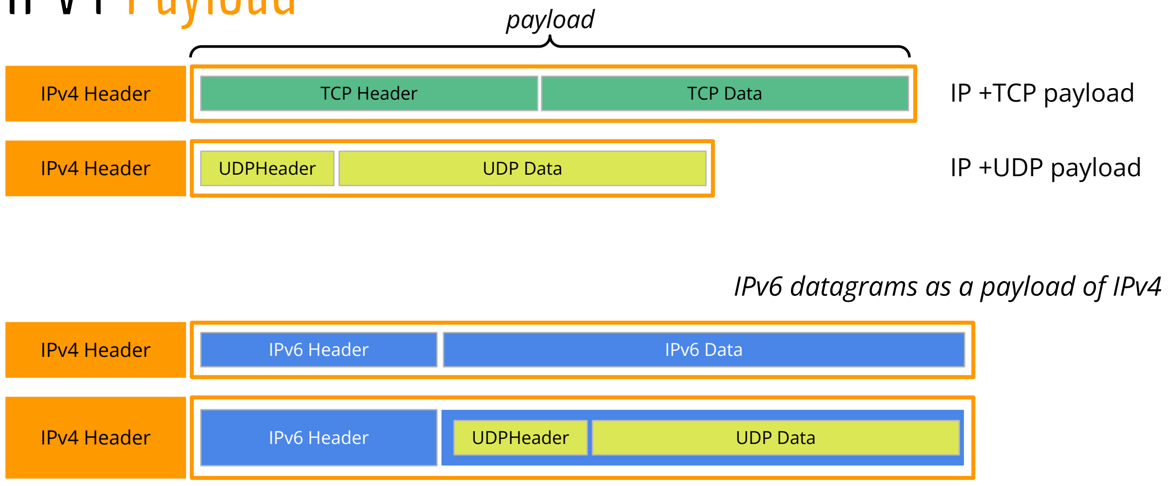
IPv4 & IPv6 Interoperability

Sender Address	Receiver Address	Communicate?	Explanation
IPv4	IPv6	Possible	• *IPv4 host cannot initiate a connection to an IPv6 host* • *IPv4 host can respond AFTER a connection is established by an IPv6 host (using NAT-64)*
IPv6	IPv4	Embed IPv4 address as IPv6 address	Use Dual Stack, Tunnelling, or NAT-64

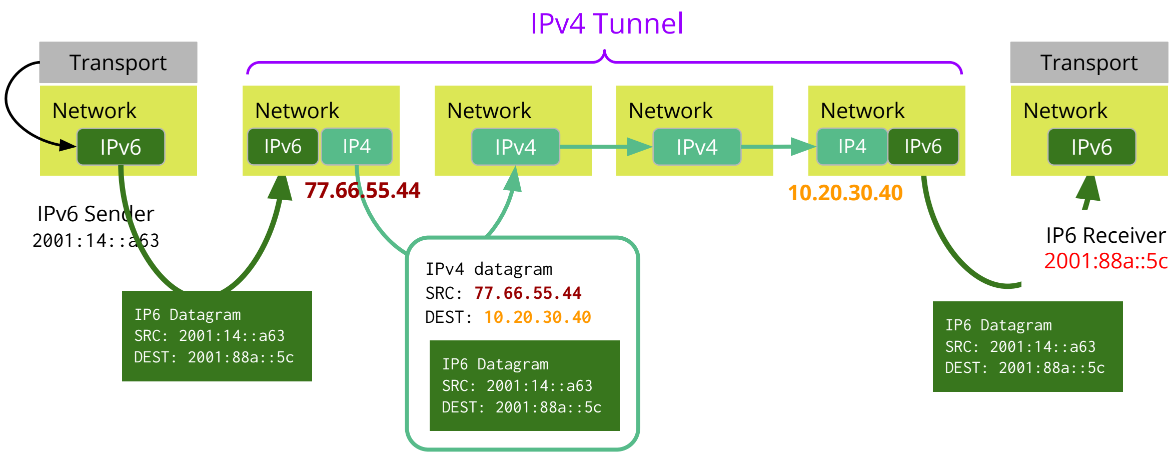
Dual Stack: Network Layer does both IPv4 and v6



IPv4 Payload



Tunnel: IPv6 Transmission via IPv4 Router(s)

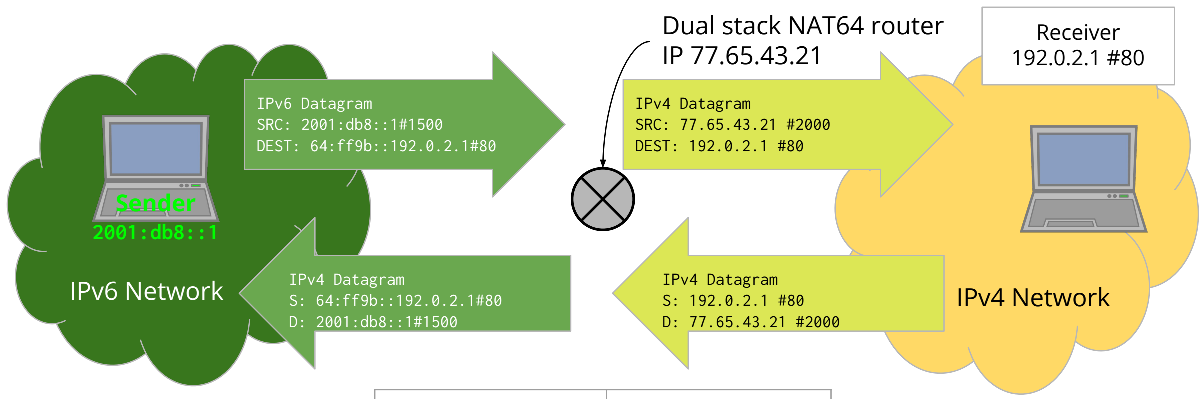


IP6 datagram is wrapped as data inside IPv4 datagram

NAT-64

- Network Address Translation
  - From IPv6 to IPv4 (and vice versa)
- Documentations
  - RFC6052: IPv6 Addressing of IPv6/IPv4 Translators
  - RFC6146: Stateful NAT64 Network Address & Protocol Translation from v6 client to v4 server

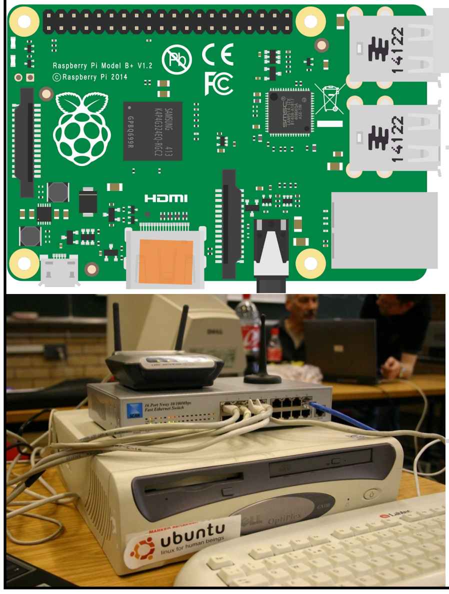
NAT64 (Net Addr Translation IPv6 to IPv4)

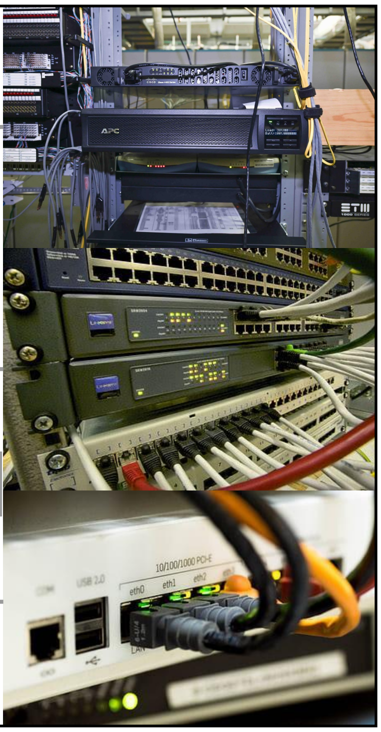


IPv6 Address	IPv4 Address
2001:db8::1#1500	77.65.43.21 #2000

Generalized Forwarding
Using *Software Defined Network* Layer

Router Hardwares





Old Graphics Cards vs. Modern GPUs

Fixed Graphics Pipeline

Programmable Graphics Pipeline

Old Routers vs. New Routers

Fixed Forwarding Logic

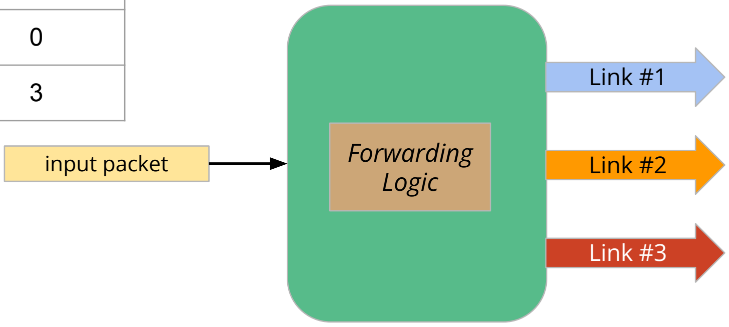
Programmable Forwarding Logic

Packet Forwarding

Old Technique

Address Prefix	Output Port
1000 1100 01xx xxxx	2
1101 101x xxxx xxxx	0
1101 1010 1001 xxxx	3

OutputLink = F (InputPacket)



Generalized Forwarding

- Traditional routers (hardware) are designed to have only “fixed address lookup” function
    - Map destination address contained in the incoming packets to output link to forward the packet
- (Re)configuring these routers are typically accomplished by executing CLI commands specific to the router manufacturer
    - Tedious task to reconfigure hundreds of router in a huge data center
- Newer routers are designed to be more programmable
    - Similar idea to old days of Fixed Functions Graphics Pipeline vs. Programmable Graphics Pipeline (in GPUs) today

Motivation: Borrow Ideas from Modern GPUs

Traditional Graphics Pipeline

- Actions performed by each stage of the pipeline are fixed

Modern Graphics Pipelines (GPUs)

- Actions performed by some stage can be customized by a *shader* program (vertex shader, geometry shader, fragment shaders)

Traditional Routers (hardware)

- Perform address lookup and forward incoming packets to one of the output links

Modern Routers

- Can perform more actions besides only forwarding packets
- Standard: OpenFlow

Generalized Forwarding (Match + Actions)

	Traditional Routers	Modern Routers
Functionality	Fixed Address Lookup functions (**Match** Address in the LUT)	Programmable Packet Matching
Actions on Packet	Forward to Output Link	Many other **actions**: **drop, modify, copy, log, prioritize, rewrite headers**
Analogy	Fixed Graphics Pipeline	Programming Graphics Pipeline (modern GPUs)
Standard		OpenFlow

Software Defined Network
(Software Defined Network Layer)

OpenFlow: Match

- By Properties from the Link Layer
  - MAC Address (Medium Access Control
  - Virtual LAN ID, Virtual LAN priority
- By Properties of the Network Layer
  - IP address (source/destination)
  - IP protocol type
- By Properties of the Transport Layer
  - Port number (source/destination)

OpenFlow: Actions

- Forward packet
- Drop packet
- Log packet
- Modify-Field