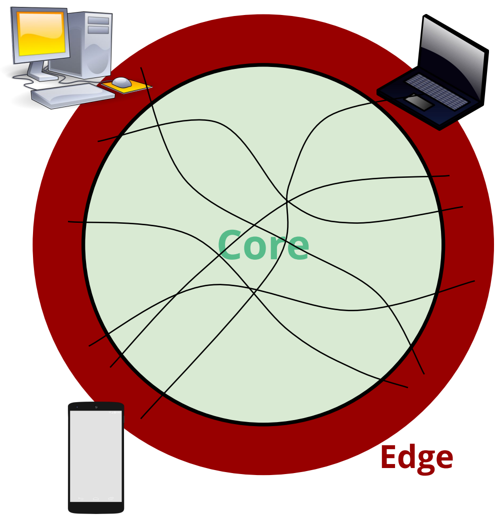
Chapter 04
Network Layer

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



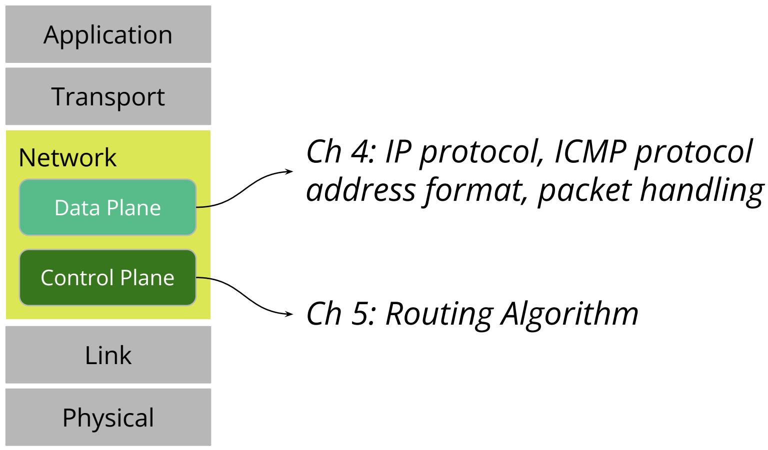
Core: interconnected routers

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**

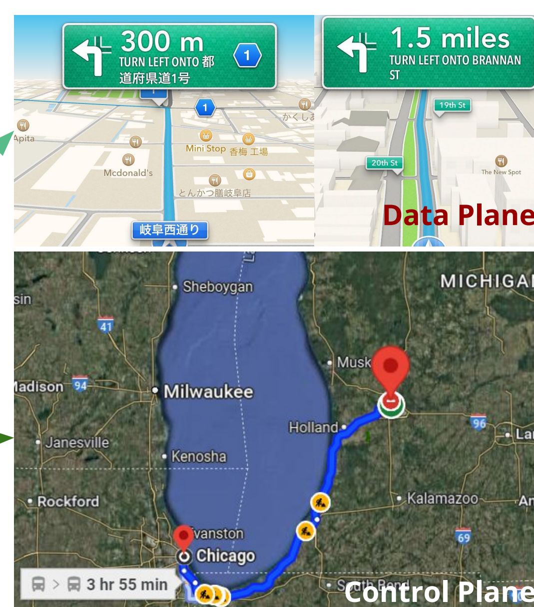
Network Layer: Data and Control Planes



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!

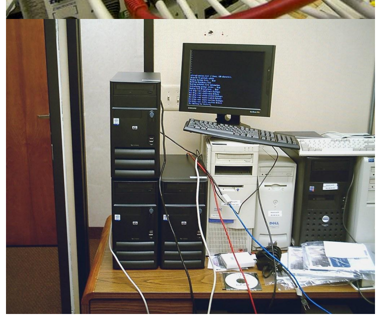
Primary Jobs of the Network Core (Recall Ch01)

- 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

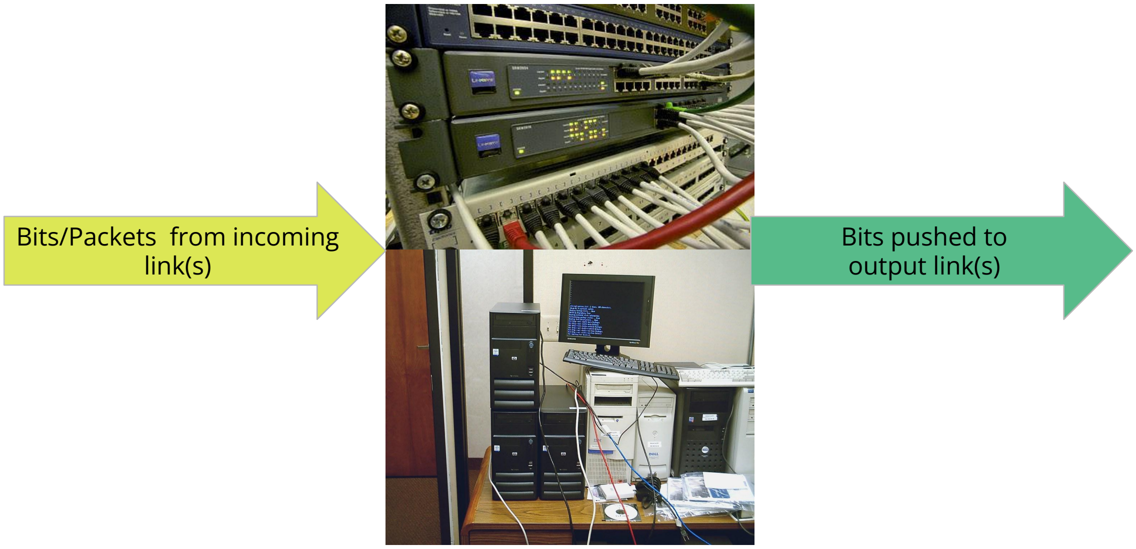
Packet Forwarding by a Router

Dedicated Router Hardware

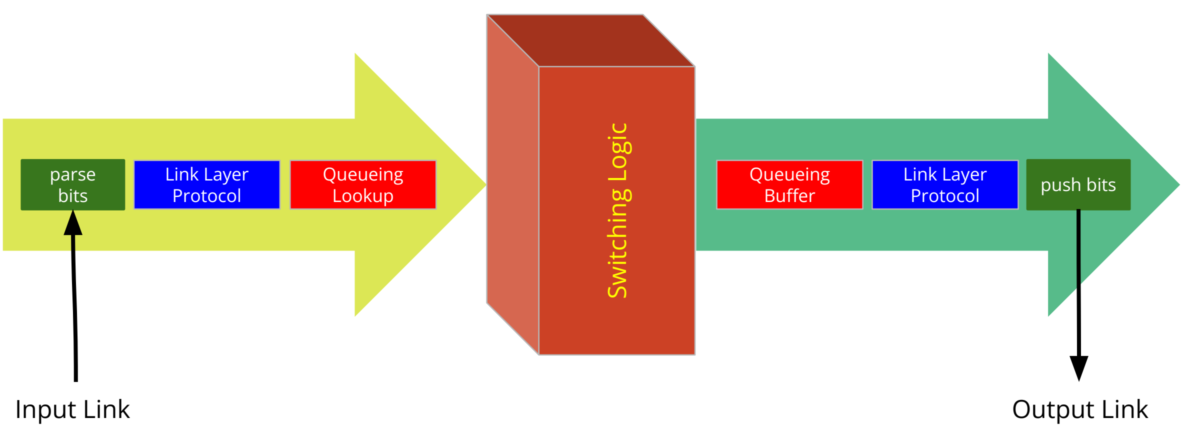


General Purpose Computer

Packet Forwarding by a Router

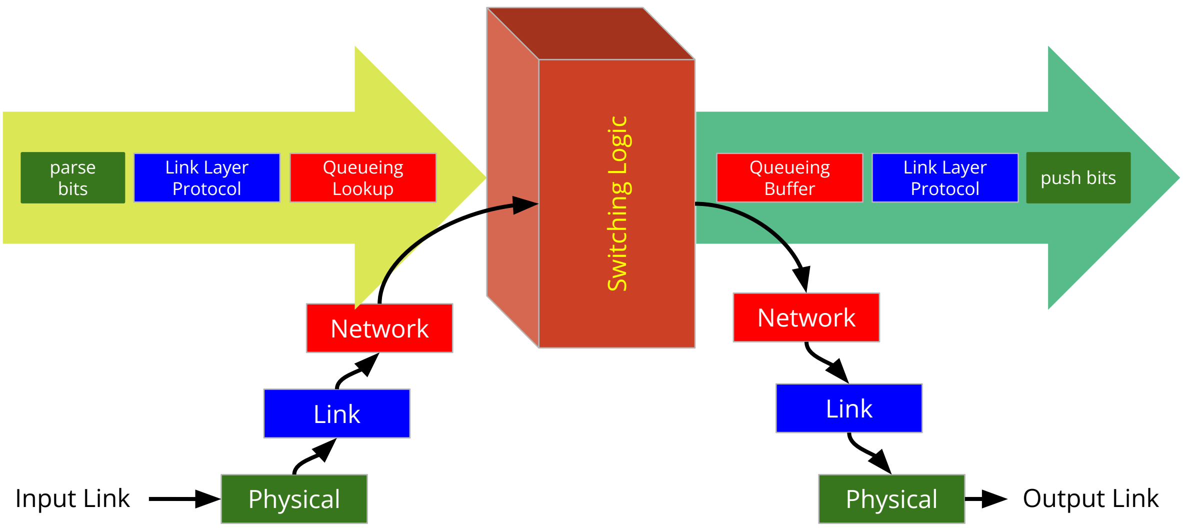


Router Architecture



Also: Kurose & Ross Slide #14

Router Architecture



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, ...)

Destination-Based Forwarding

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

See also Kurose & Ross slides page 17-29

Routing Table: Address Range vs. Address 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

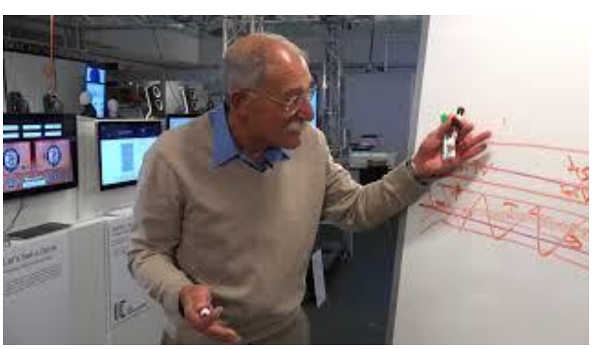
Address Prefix	Output Port
1000 1100 01xx xxxx	2
1101 101x xxxx xxxx	0
1101 1010 1001 xxxx	3

Packet Scheduling Policy

- FCFS
- **Priority**
  - Based on what fields/packet properties?
- Round Robin
- Weighted Fair Queueing (Generalized Round-Robin)

Also: Kurose & Ross Slides 35-38

Net Neutrality



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?

Packet Drop Policy (When buffer is full)

- Tail drop: drop the newest arrival
- Priority drop: drop the lowest priority
- Retain special packets: packets which notify congestion

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

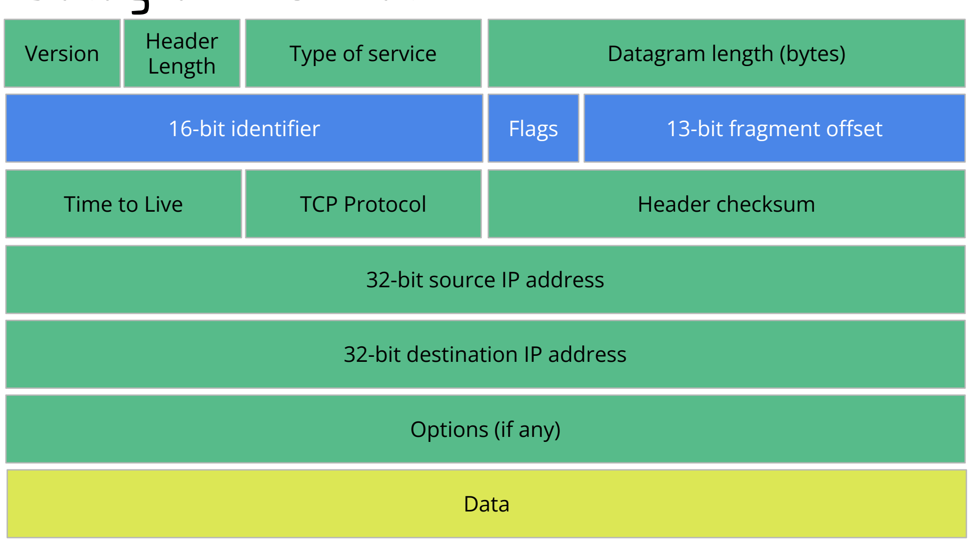
$$\text{Buffer Size} = \frac{RTT \cdot C}{\sqrt{N}}$$

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



IPv4 Address: binary & dot notation

gvsu.edu   104.17.87.18

0110 1000	0001 0001	0101 0111	0001 0010
104	17	87	18

IPv4 Address Range (CIDR Notations)

104.17.91.0 – 104.17.91.255 ⇒ Subnet 104.17.91/24

same 24 bits

0110 1000	0001 0001	0101 1011	0000 0000
104	17	91	0

0110 1000	0001 0001	0101 1011	1111 1111
104	17	91	255

same 22 bits

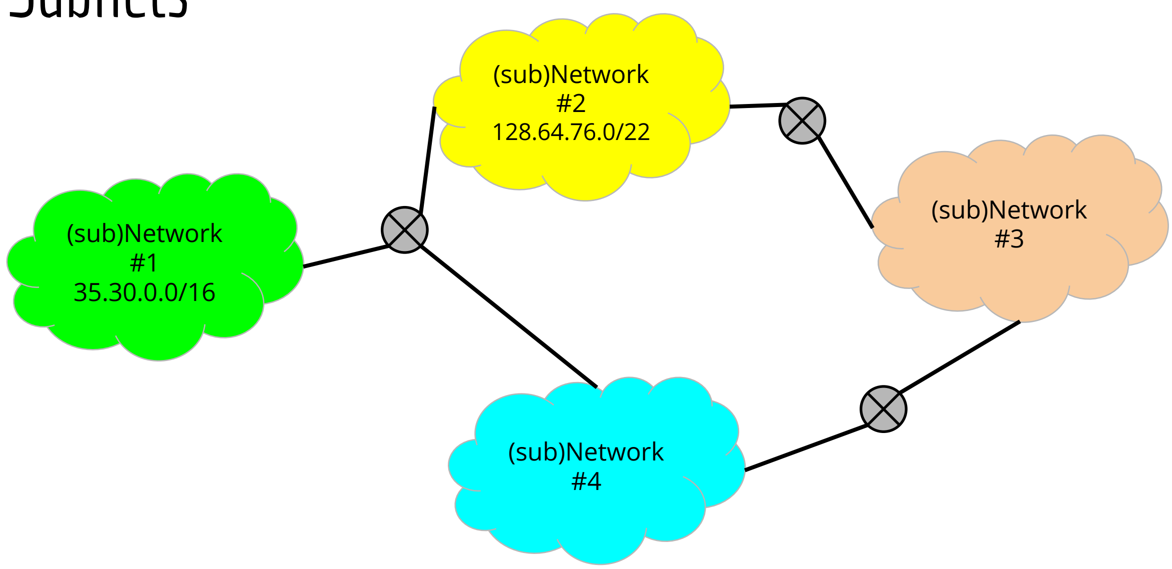
0110 1000	0001 0001	0101 1000	0000 0000
104	17	88	

same 22 bits

0110 1000	0001 0001	0101 1011	1111 1111
104	17	91	

104.17.88.0 – 104.17.91.255 ⇒ Subnet 104.17.88/22

Subnets



Broadcast Addresses

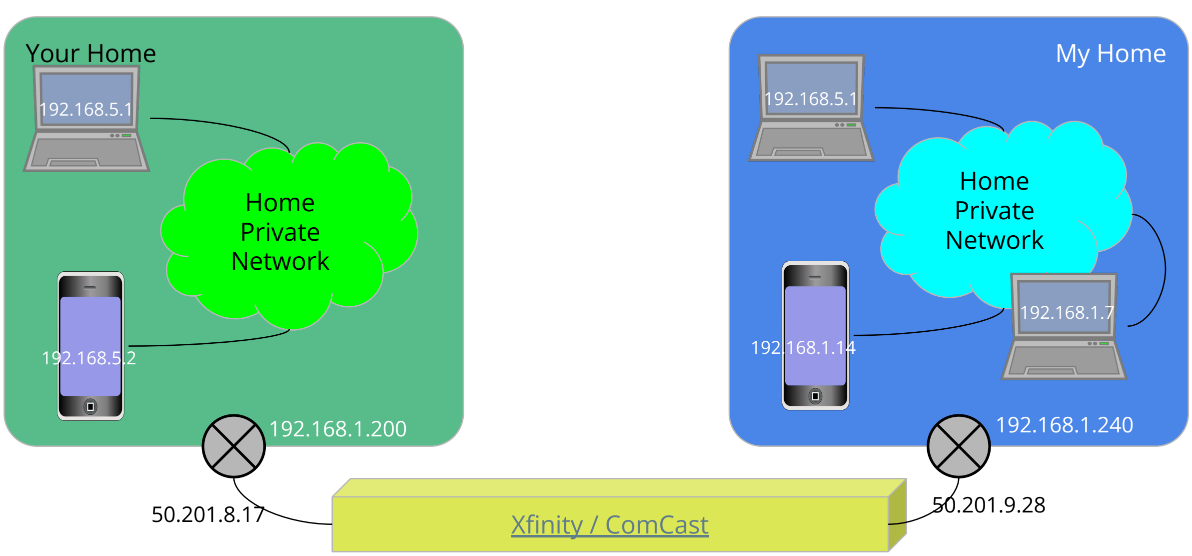
	IP Address	Forwarded to other subnets
Local Broadcast	255.255.255.255	No
Directed Broadcast	Last/Highest address in a subnet	Yes

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

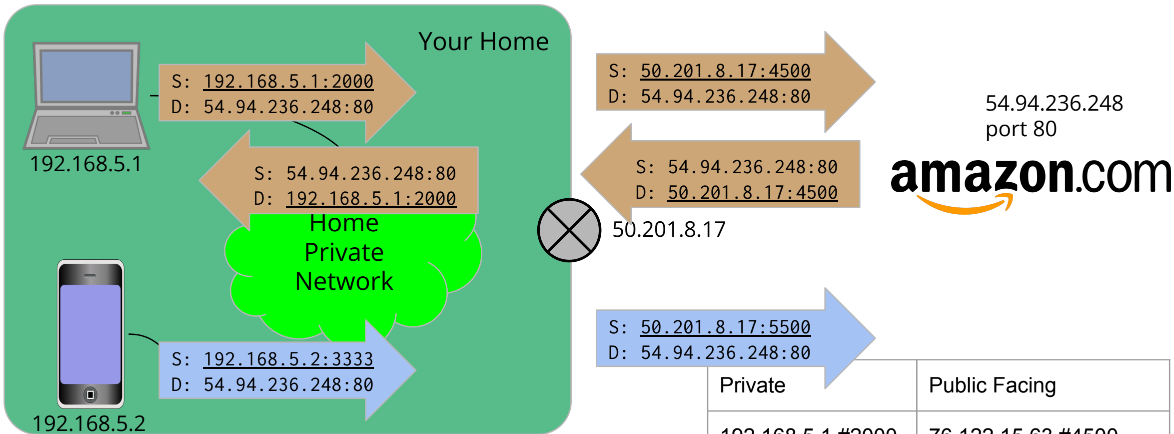
Private IP Addresses

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	2^{24} - 2 hosts
Class B	172.16.0.0/12	10101100 0001xxxx yyyyyyyy zzzzzzzz First byte: 0xAC, underlined is a ‘B’	20	2^{20} - 2 hosts
Class C	192.168.0.0/16	11000000 10101000 yyyyyyyy zzzzzzzz First byte: 0xC0	16	2^{16} - 2 hosts

Private IP Addresses



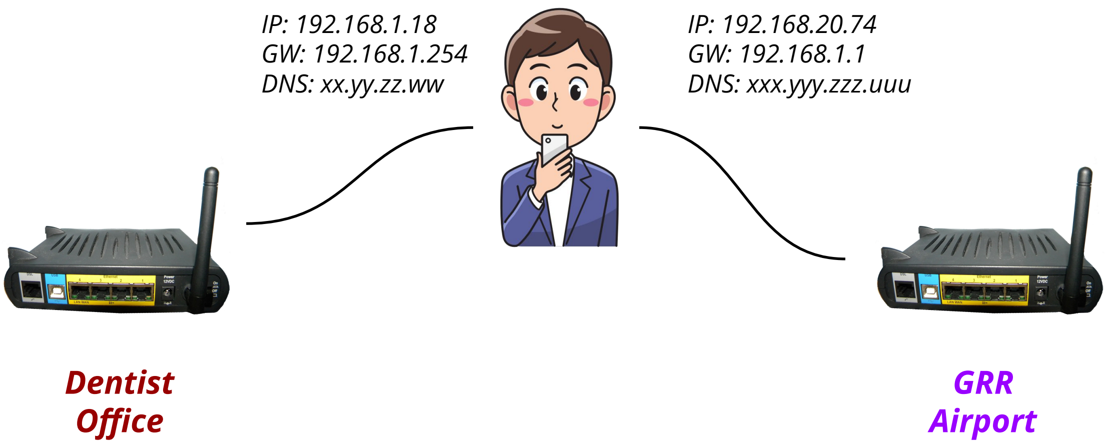
NAT: Network Address Translation



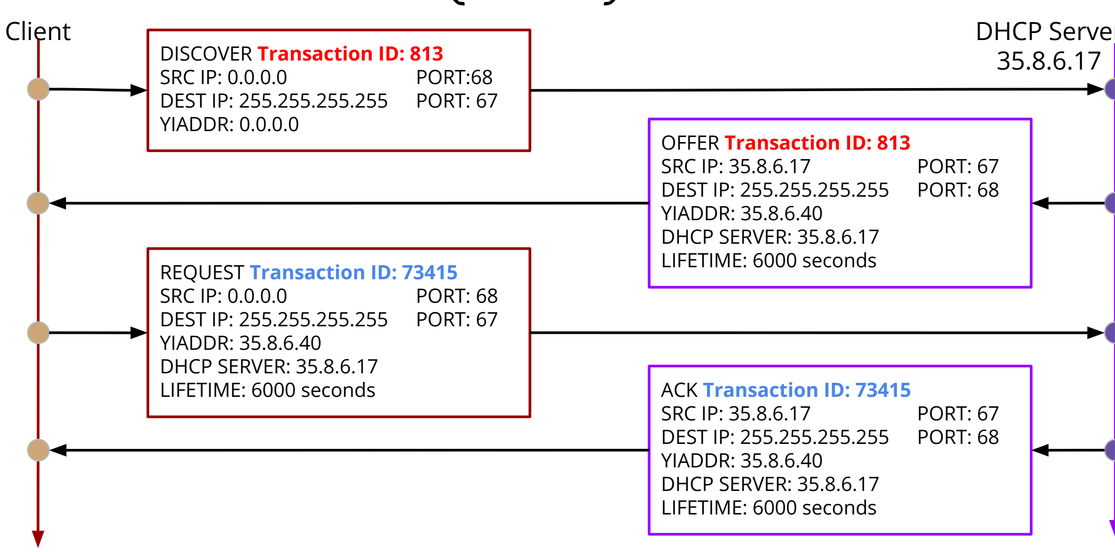
Private	Public Facing
192.168.5.1 #2000	76.122.15.63 #4500
192.168.5.2 #3333	76.122.15.63 #5500

Dynamic Host Configuration Protocol (DHCP)

Connecting Your Phone/Laptop To (Home) WiFi

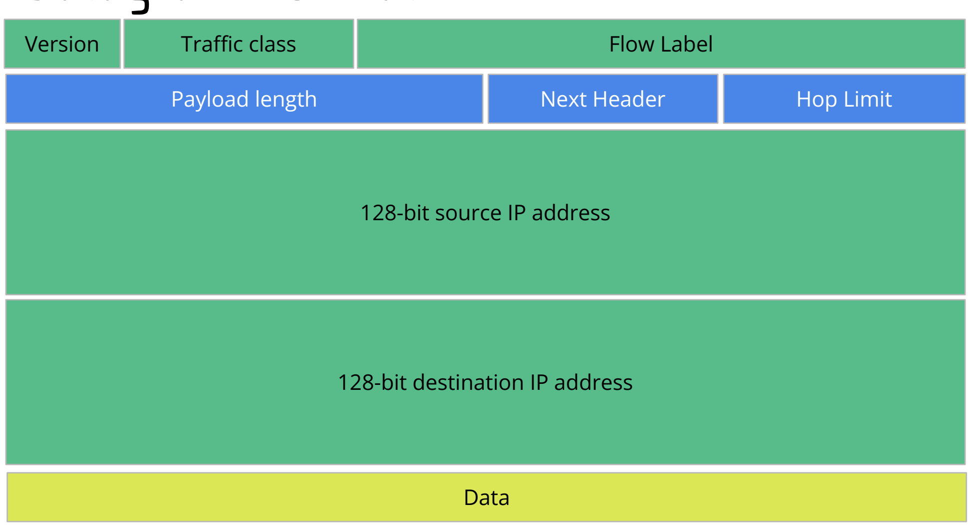


DHCP Transactions (DORA)



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)

ABCD:**EF01**:2345:6789:**AbcD**:ef01:2345:6789

0111 1111 0000 0001

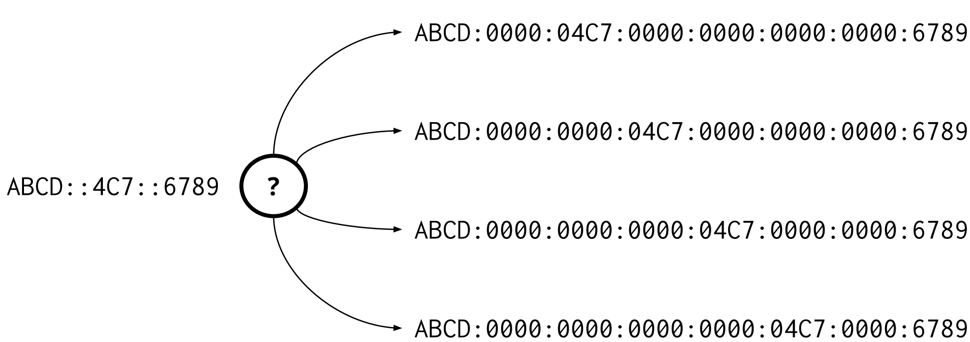
1010 1011 1100 1101

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



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)
2001:DB8::/32	Global Unicast Address??
2000::/3	Global Unicast Address (Public Unique Addresses)
64:FF9B::/96	NAT64 prefix
FE80::/10	Link-Local Unicast
FC00::/7	Unique Local (Similar to Private IP addresses in IPv4)
FEC0::/10	Site-Local Unicast (deprecated)
FF00::/8	Multicast (RFC4291)

IPv4 address embedding example (RFC6052)

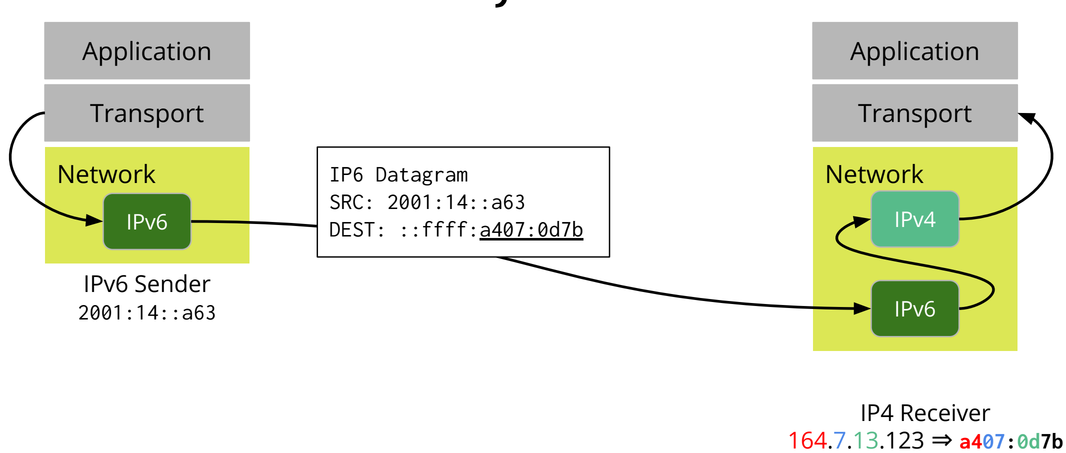
IPv4: 192.0.2.33 (in hex C0.0.2.21 ⇒ C000:221)

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

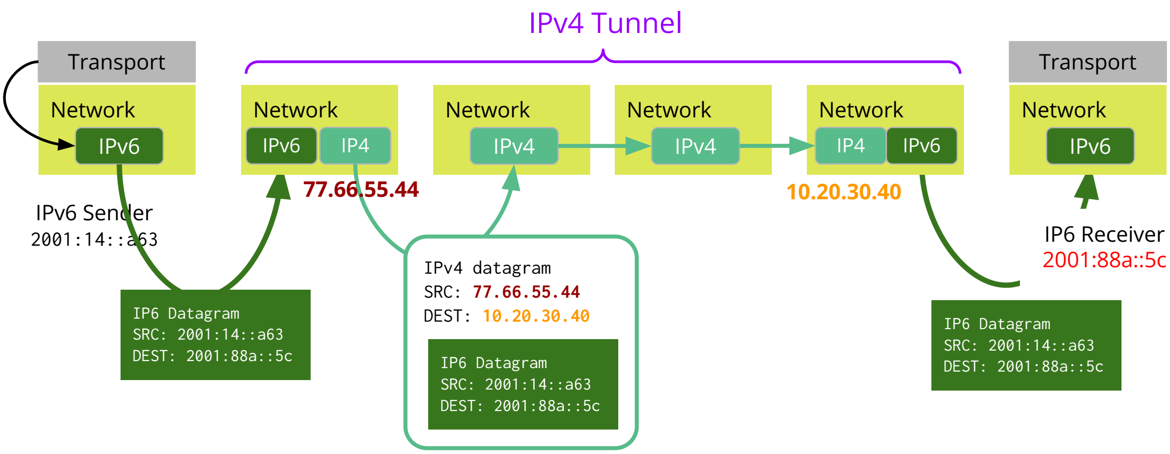
IPv4 & IPv6 Interoperability

Sender Address	Receiver Address	Communicate?	Explanation
IPv4	IPv6	Possible	• *IPv4 host cannot initiate a connection to an IPv6 host* • *IPv4 host can respond to a connection initiated 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 IP v4 and v6



Tunnel: IPv6 Transmission via IPv4 Router(s)

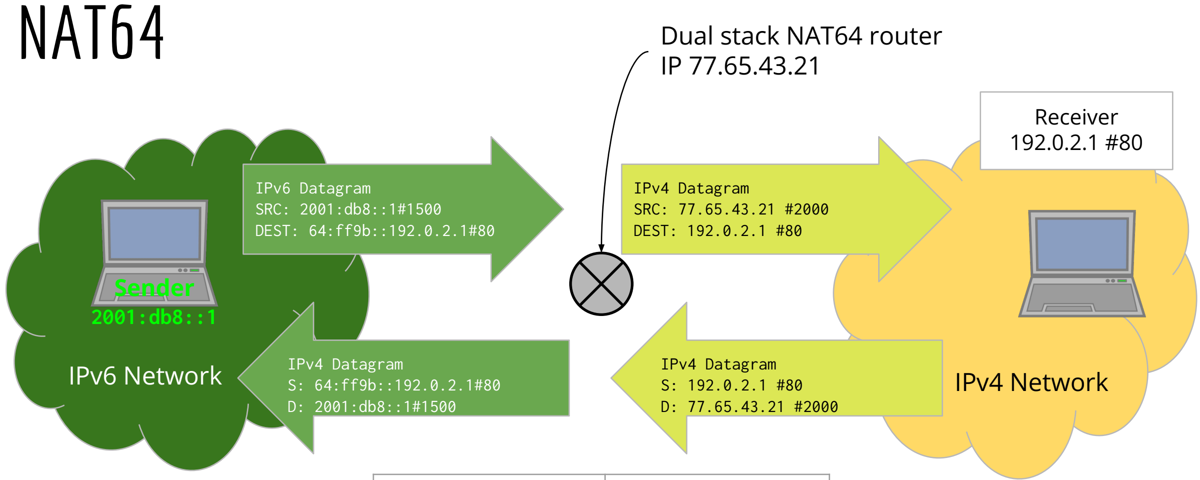


IP6 datagram is encapsulated as data inside IPv4 datagram

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

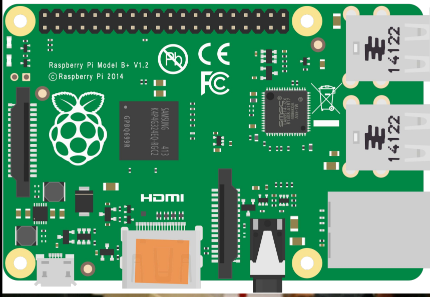
NAT64



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

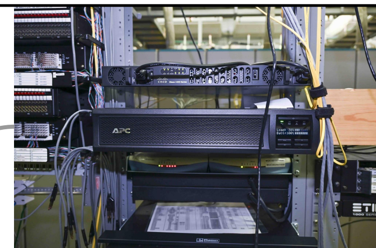
Generalized Forwarding
Using *Software Defined Network* Layer

Router Hardwares

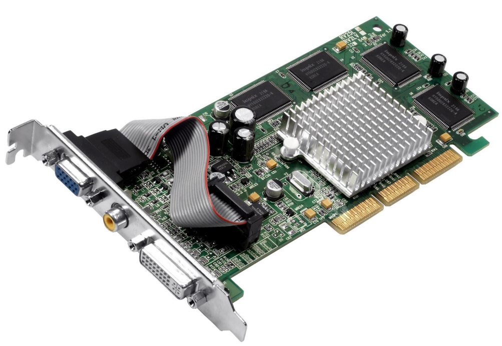


Ordinary Computers

Specialized Hardware



Old Graphics Cards vs. Modern GPUs



Fixed Graphics Pipeline

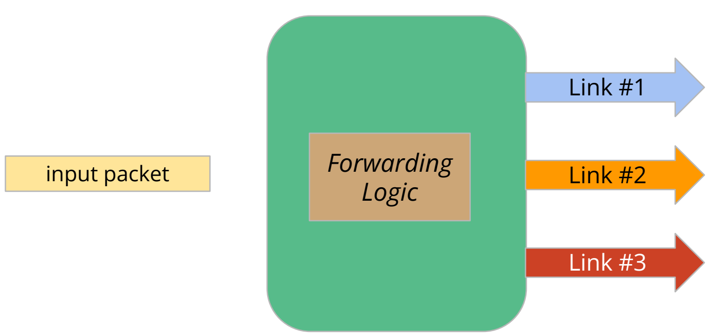
Programmable Graphics Pipeline

Old Routers vs. New Routers

Fixed Forwarding Logic *Programmable Forwarding Logic*

Packet Forwarding

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