Addresses, Routing and Control

Unit 10 - Hands-On Networking - 2018

Prof. Dr.-Ing. Thorsten Herfet, Frank Waßmuth, Andreas Schmidt, Pablo Gil Pereira

Telecommunications Lab, Saarland Informatics Campus, 22nd Feb. 2018

Recap

- Transport Layer
  Enables communication between processes.
  - **RDT** Reliable Data Transport
  - **UDP:** Unreliable Datagram Transport
  - **TCP:** Reliable Stream Transport
  - **PRRT:** Multimedia transport with ARQ/FEC

❓ How is data actually transmitted between hosts?

❓ What's an IP address?


Application


Presentation


Session

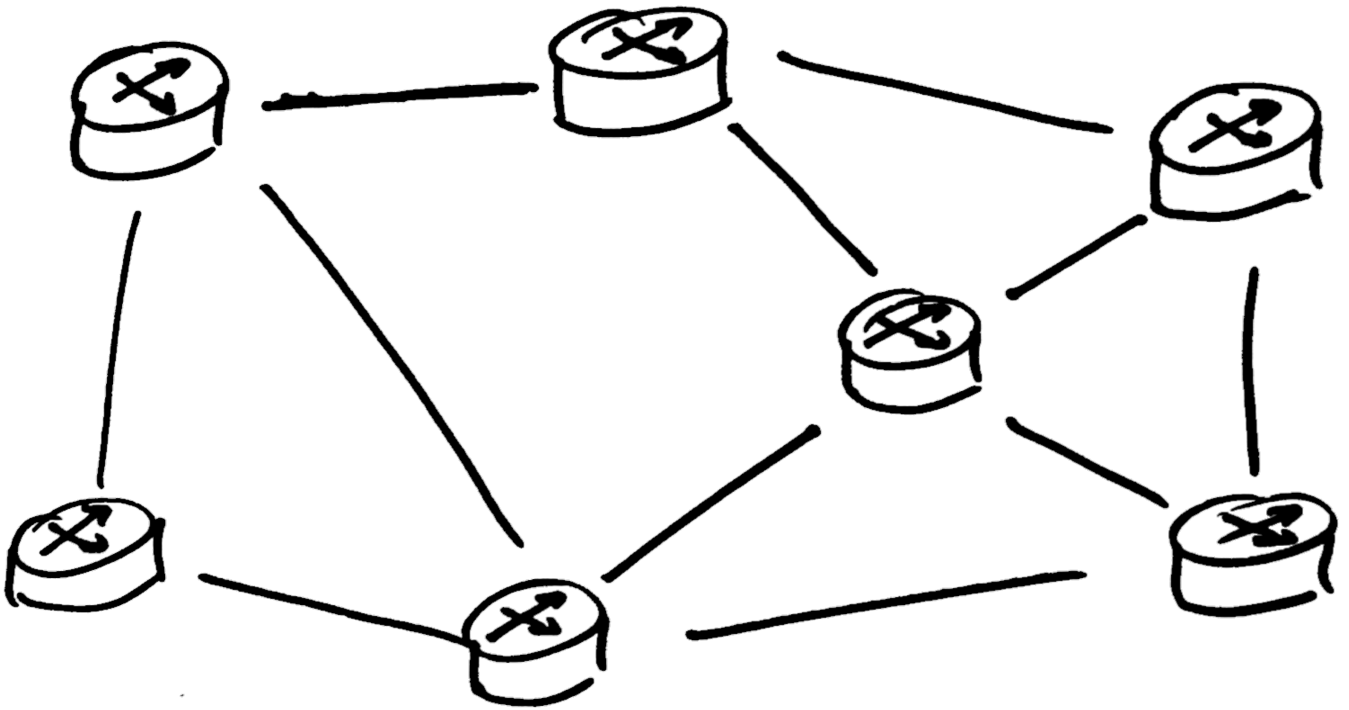

Transport


Network

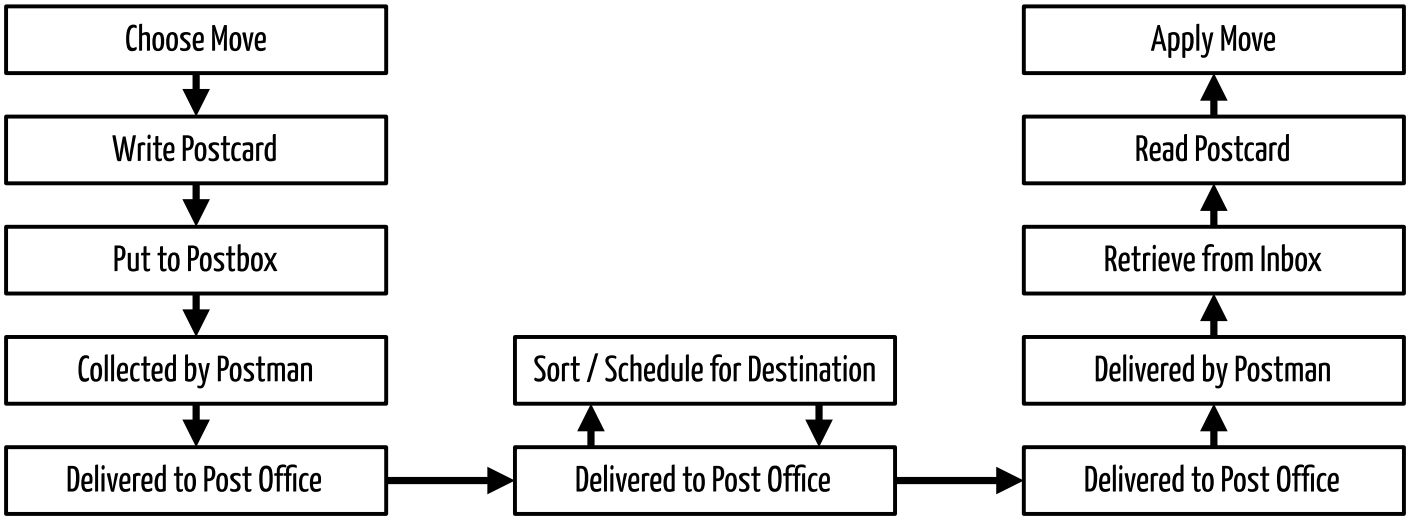

Link


Physical

A Typical Network



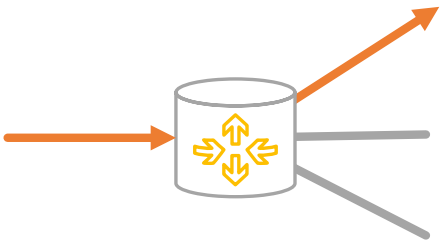
Going Postal … with Chess



Network Layer Functions

Forwarding

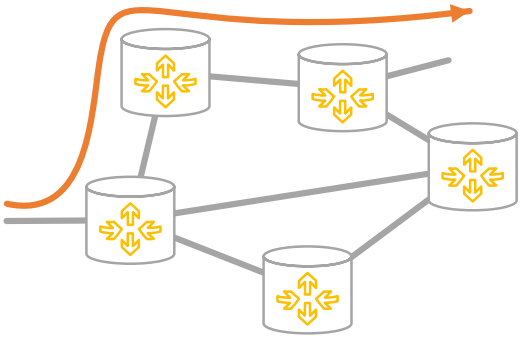
- Move data from an input link to an output link.



- Node-level process of moving data from one link to another

Routing

- Find out to which output link data needs to go.



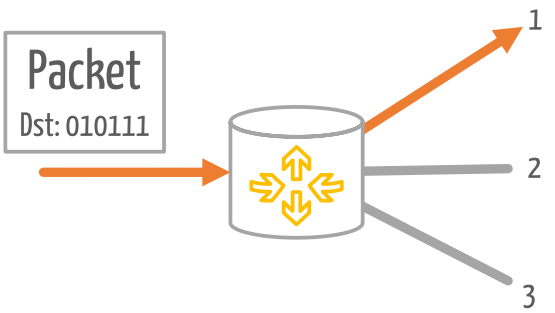
- Network-level process to determine the of the data.

A device forwarding data from one link to another is typically called a **Router**. 😐

The Forwarding Table

- Lookup table found in every network device
- Describes topology of the network immediately around the device
- Keys **destination** to **output link**

Destination / VC	Output Link
010110	1
010111	1
011100	2
111110	3



Routing

Routing is responsible for filling the forwarding table with sensible information.

Depending on the method used to fill the forwarding table we differentiate different types of routing.

Static Routing

- Forwarding tables are propagated manually

Dynamic Routing

- Forwarding tables are propagated automatically by using specialized routing protocols.
  (**Distributed Routing**)
- Forwarding tables are propagated by a central coordinator.
  (**Centralized Routing**)

Quiz

Which services are provided by the Network Layer?

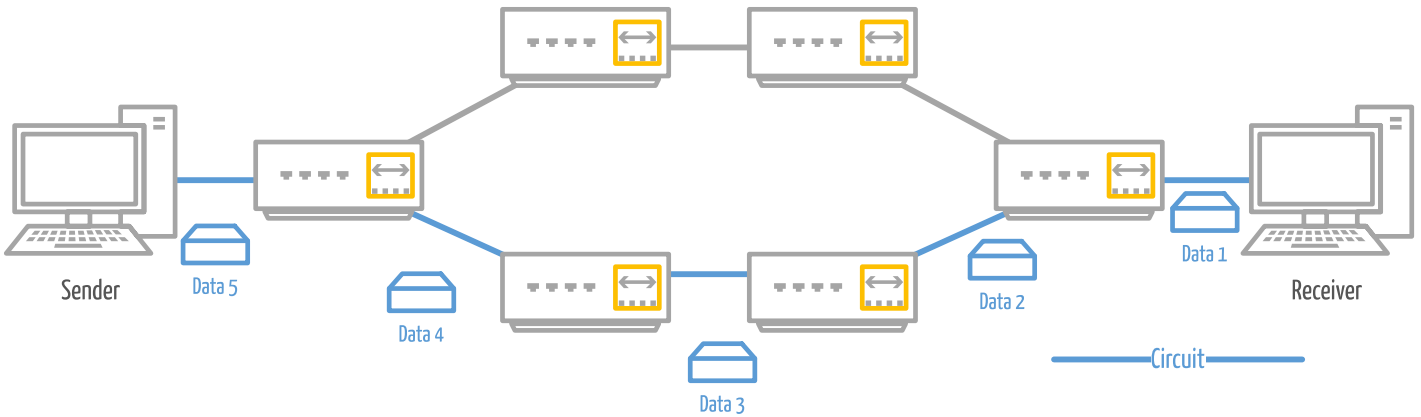
- Guaranteed Delivery
- Bounded-delay Delivery
- Guaranteed Minimal Data Rate
- In-order Packet Delivery
- Guaranteed Maximum Jitter
- Security Services

Answer:

It depends.

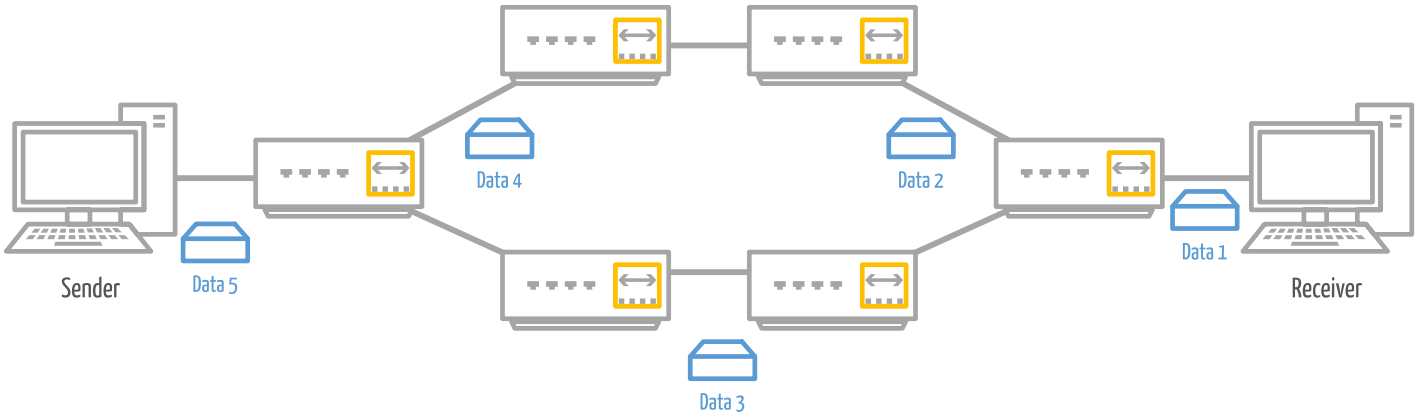
Virtual-Circuit Networks

Idea: Set up connections through the network layer before exchanging data.



- Packets are tagged with the **VC ID** (not the destination)
- Routers maintain **connection state**, all packets take the same route
- (Optional) Resource Allocation (minimum data rate)
- In-Order Delivery, (Near) Constant Delay

Datagram Networks



- Each packet is tagged with the address of the destination.
- Routers do **not** maintain **connection state**
- No data rate / (in-order) delivery / timing guarantees (**Best-effort**)

VC vs Datagram

Virtual-Circuit

- Built for **voice transmission**
- **"Dumb"** end-systems (e.g. rotary phones)
- **High complexity** in the network

Datagram

- Built for **data communication**
- **"Sophisticated"** end-systems (i.e. computers)
- **Less complexity** in the network

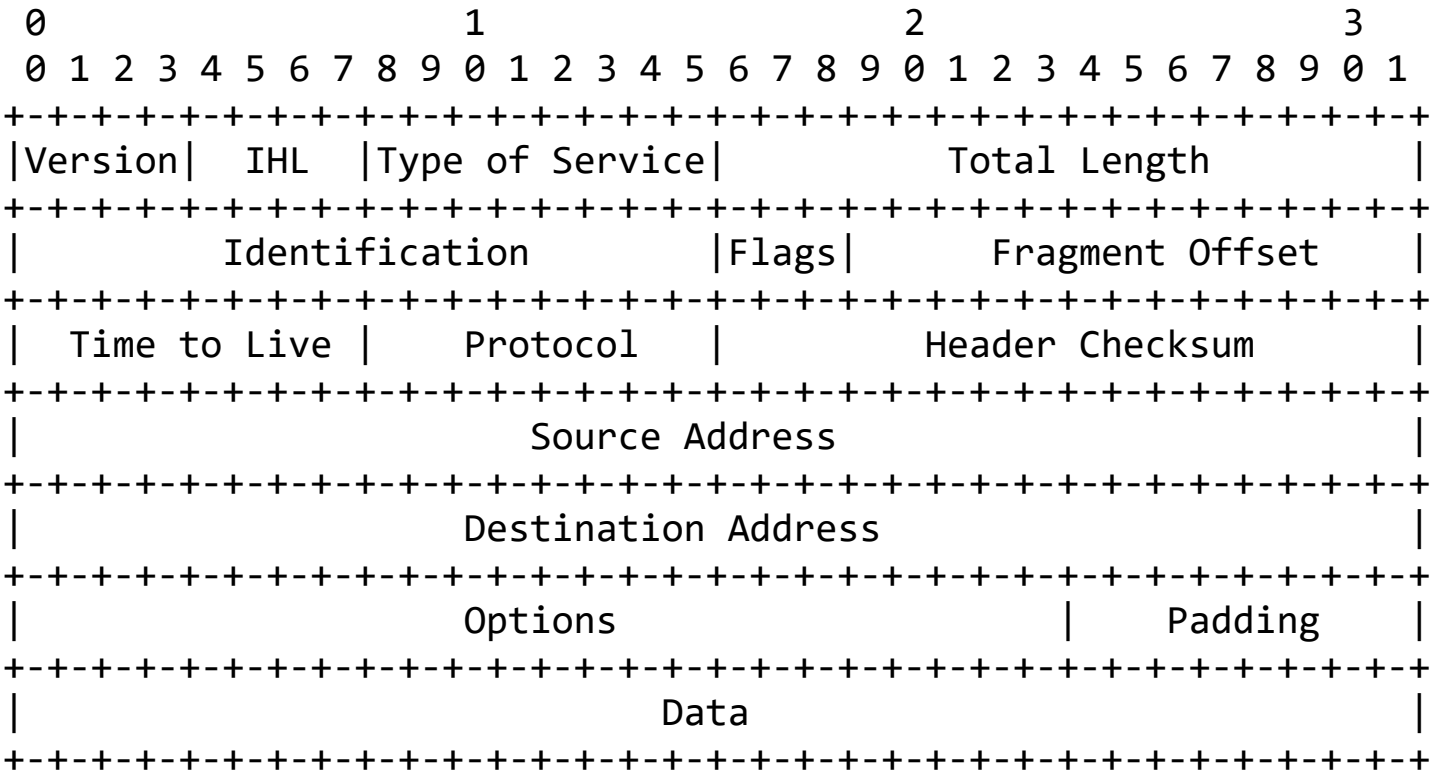
The Internet Protocol (Version 4)

- Defined in **RFC791**
- Dominant network layer protocol of the Internet
- **32 bit** addresses
- Referred to as **IPv4**
- Provides **connectionless** Datagram Service
- An IP datagram is called a **packet**

Example Addresses and Notation:

Decimal (Dotted Quad)	Binary	Hex
134.96.1.69	10000110 01100000 00000001 01000101	0x86600145
192.168.1.2	11000000 10101000 00000001 00000010	0xC0A80102
224.0.0.1	11100000 00000000 00000000 00000001	0xE0000001

IPv4 | Datagram Format



IPv4 | Header Fields

Version (4 bit):
IP protocol version (It's 4).

IP Header Length (4 bit):
Specifies the length of the **header** in bytes.
The typical header size (i.e. without any options) is 20 bytes.

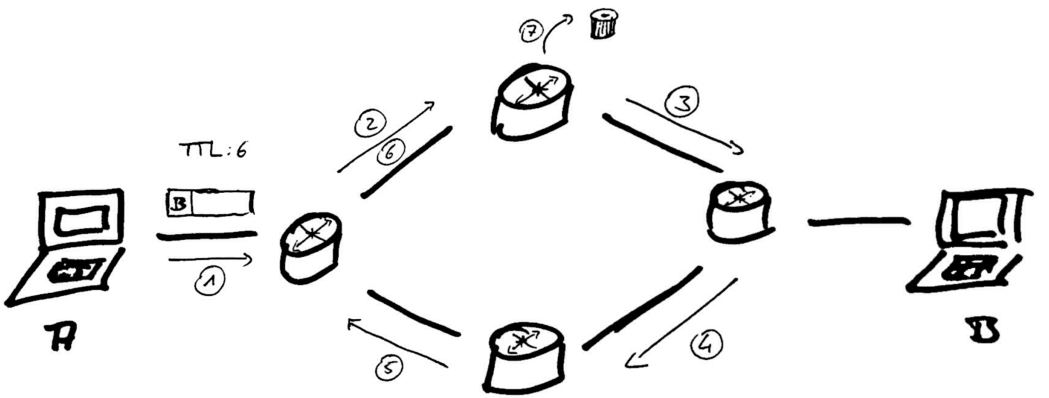
Type of Service (8 bit):
TOS bits are a way to distinguish different types of datagrams. These can be used to **suggest** e.g. low-delay-, high-throughput- or more reliable transport. Whether or not datagrams actually get special treatment is up to the administrator of the router.

Total Length (16 bit):
The length of the **complete datagram** (header + data) in bytes.

ID, Flags, Offset (16, 3 and 13 bit):
These three fields are used to fragment datagrams in case they are too big for the underlying link-layer architecture.

IPv4 | Header Fields

Time-to-live (8bit):
Also called the **TTL** or **hop limit**, the TTL makes sure that packets do not stay in the network forever. This might happen when there are badly configured forwarding tables, causing a packet to run in circles. The value of this field is decremented by one each time a router processes the datagram. If the TTL reaches **0** the datagram is discarded.



IPv4 | Header Fields

Protocol (8bit):

The transport layer protocol encapsulated in this datagram. This field is usually only used by end-hosts to determine which transport-layer protocol should handle the datagram's contents.

Number	IP Protocol
6	TCP
17	UDP

(see IANA Assigned Internet Protocol Numbers for a complete list).

Checksum (16 bit):

A checksum calculated over the **header contents**. A router computes and compares this checksum for every datagram it receives. If the sums are **not equal**, the datagram is usually **discarded**. A packet's checksum has to be **recalculated and updated in the header** at every router.

IPv4 | Header Fields

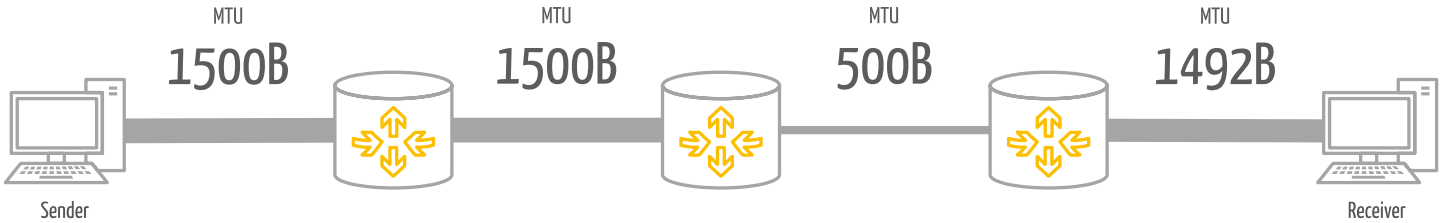
Source / Destination Address: (32bit each)
When a source creates a datagram, it inserts its IP address into the source address field and inserts the address of the destination into the destination address field.

Options:
The IP datagram header can be extended using options. Header options were meant to be used rarely and therefore not included in the fixed header parts. The existence of options however already makes things more complicated, since the header does not have a fixed length and some options need to be inspected by routers while others don't. Processing time per datagram may vary, which can be a problem for high-performance routers and hosts.

Data:
The data payload of the datagram. The type of data is described by the protocol field.

IPv4 | Fragmentation

Spoiler Alert: Not all link-layer protocols can carry datagrams of the same size!



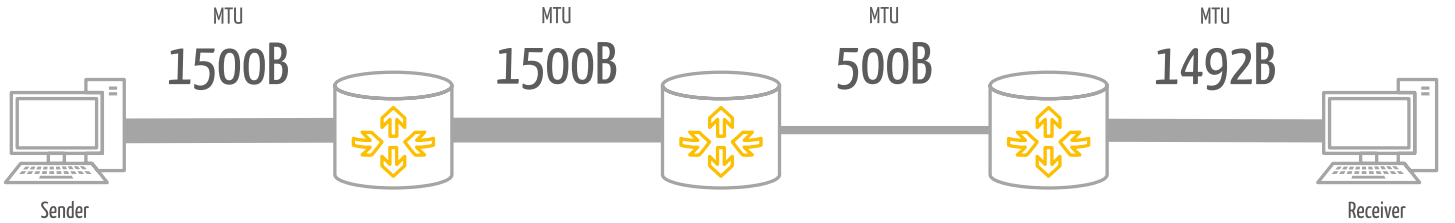
Maximum Transmission Unit

The **MTU** of a link is the size of the largest datagram the link layer can pass on. It is given in bytes.

The smallest MTU on a path from a source to a destination is called the **Path MTU**.

IPv4 | Fragmentation

💣 **Spoiler Alert:** Not all link-layer protocols can carry datagrams of the same size!



- Packets **bigger than the MTU of the next link** segment need to be split up into multiple **fragments**.
- Fragmented packets have to be **reassembled again for the transport layer**.
- **Intermediate routers fragment** packets, but **only end-hosts reassemble**.

IPv4 | Fragmentation

The IP header includes three fields only used for fragmentation.

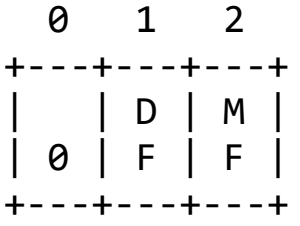
ID

"Unique" ID of the IP datagram.
Shared by all fragments.

Fragmentation Offset

Indicates where in the data the fragment belongs. The offset is given in chunks of **8 bytes**.

Fragmentation Flags



- Bit 0: reserved, must be zero
- Bit 1: (DF)
  - 0 = May Fragment
  - 1 = Don't Fragment.
- Bit 2: (MF)
  - 0 = Last Fragment
  - 1 = More Fragments.

Quiz

A router receives a packet with a size of **1500 bytes**. The MTU of the outgoing link is **500 bytes**.

How many packets will the router send out?

Answer:

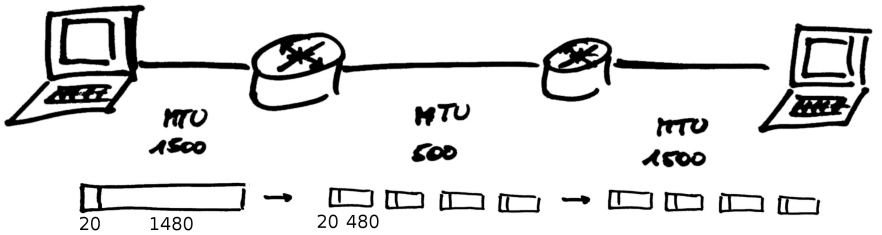
✔ The router will send out 4 fragments.

Why?

✔ The original packet contains 20 bytes of header data, so will each of the fragments.

$$\mathrm{NumberOfFragments} = \left\lceil \frac{\mathrm{PacketSize} - \mathrm{IPHeaderSize}}{\mathrm{NextLinkMTU} - \mathrm{IPHeaderSize}} \right\rceil$$

IPv4 | Fragmentation



Header Values

	Length	ID	Offset	MF
Original Packet	1500	1337	0	0

	Length	ID	Offset	MF
Fragment #1	500	1337	0	1
Fragment #2	500	1337	60	1
Fragment #3	500	1337	120	1
Fragment #4	60	1337	180	0

IPv4 | Fragmentation

- Enables connecting different link-layer technologies

- Complicates routers and end-systems
- Additional processing burden
- Can be used to drive lethal DoS attacks

Quiz

IPv4 uses **32bit addresses**, which yields a total of $2^{32} \approx 4.3 \cdot 10^9$ addresses.

What are the implications on the forwarding table?

Answer(s):

- ✔ Just as many entries in the forwarding table.
- ✔ Every table on the network needs to be updated if a device changes location.

This seems highly inefficent, and it is.

Longest Prefix Matching

Instead of listing every single address, define address ranges.

Range	Link
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

HON

Longest Prefix Matching

In binary ranges share a common prefix.

Range	Link
11001000 00010111 00010000 00000000 through **11001000 00010111 00010**111 11111111	0
11001000 00010111 00011000 00000000 through **11001000 00010111 00011000** 11111111	1
11001000 00010111 00011001 00000000 through **11001000 00010111 00011**111 11111111	2
otherwise	3

Longest Prefix Matching

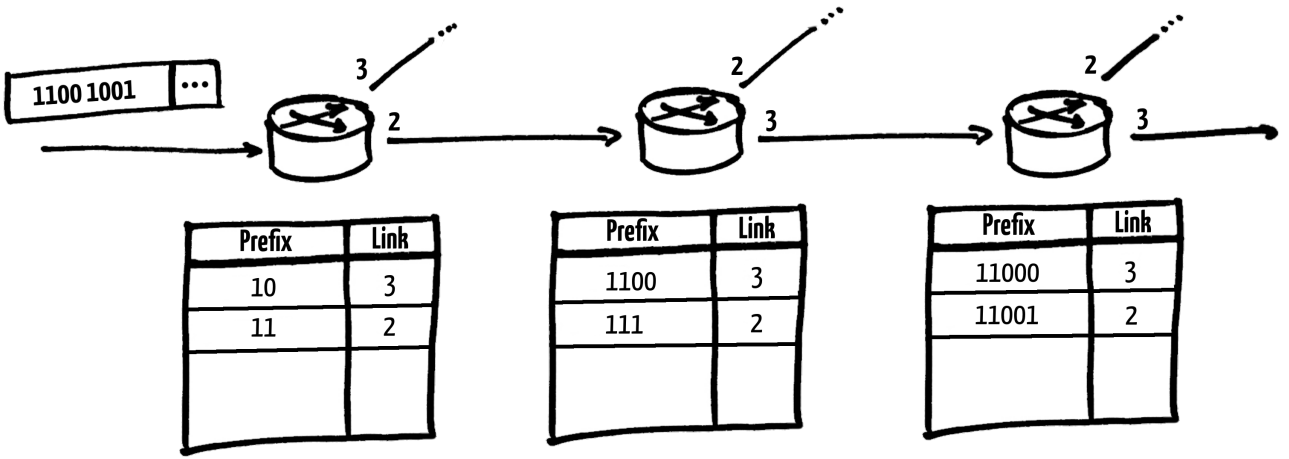
Only put the range prefix in the forwarding table.

Range	Link
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
otherwise	3

- Routers only need to look at the **longest matching prefix** to determine the output link.
- **Less entries in / changes** to the table required
- **Every prefix** represents a **different part of the network**

- In **IP Networking:**
  - Table is often simply called the **Routing Table**.
  - All addresses **sharing the same prefix** belong to the same **subnet**.

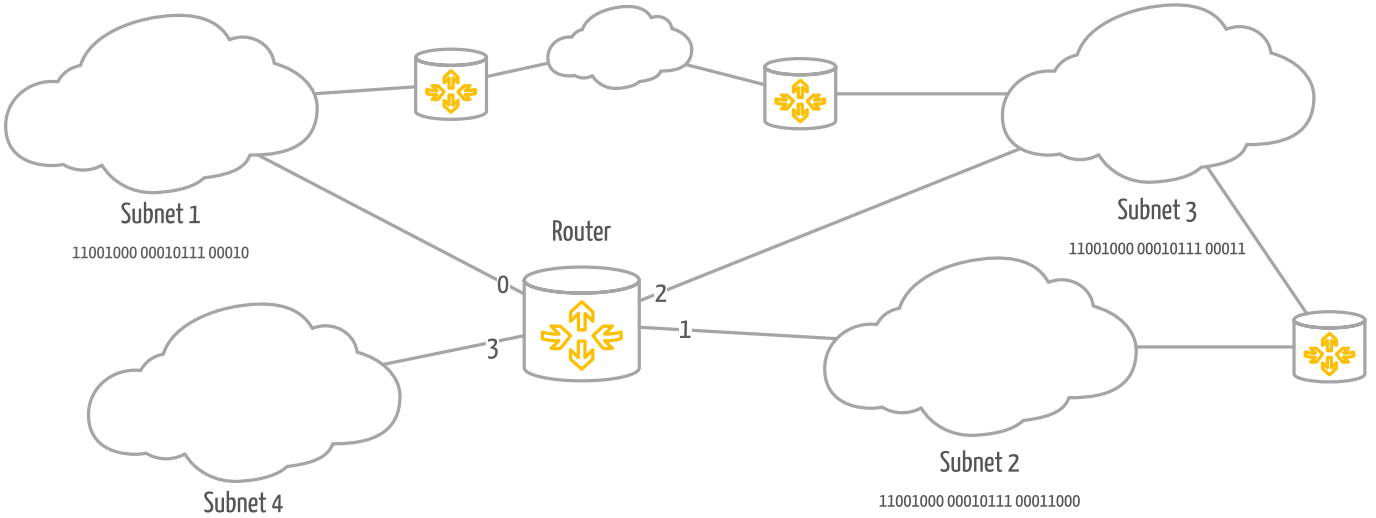
Divide and Conquer



Prefixes typically get more and more specific along the path

Called **next-hop** or **hop-by-hop routing**

IPv4 | Subnets



- The Internet is **made up of subnets** (actually: **Autonomous Systems**, see RFC1930 for more information)
- Network devices **determine whether to use a router** based on the **prefix**
  → **Every device** has to know the **prefix length of its subnet**
- Packets destined for a **non-local prefix** have to go through a **router**

IPv4 | Subnets

IP addresses are split up into a **network part** and a **host part**

CIDR Notation (Classless Inter-Domain Routing)

192.0.2.1**/24**

IP Address Prefix length

11000000 00000000 00000010 00000001
11111111 11111111 11111111 00000000

Subnet Mask Notation

192.0.2.1**/255.255.255.0**

IPv4 | Subnets

A subnet can be divided into further subnets

192.0.2.0/24

11000000 00000000 00000010 00000000

192.0.2.0/26
192.0.2.64/26
192.0.2.128/25

11000000 00000000 00000010 00000000

11000000 00000000 00000010 01000000

11000000 00000000 00000010 10000000

Multiple subnets can be aggregated to a supernet

192.0.2.0/28
192.0.2.16/28
...
192.0.2.240/28

192.0.2.0/24

Quiz

How many addresses are in the subnet 134.96.1.0/29?

A: 8

B: 16

C: 32

D: 64

Answer:

✔ A: Yes. Because $2^{32-29} = 2^3 = 8$.

✖ B: Cold.

✖ C: Colder.

✖ D: Ice Cold.

Another Quiz

How many *usable* addresses are in the subnet 134.96.1.0/29?

A: 2

B: 4

C: 6

D: 8

Answer:

A: Nope.

B: Wrong Again.

C: Yes. But why?

D: That would have been too easy, don't you think?

IPv4 | Subnets

Not all addresses in a subnet can be assigned to a host.

Network Address ("All-Zeros Address")

192.0.2.0/24 → 11000000 00000000 00000010 **00000000**

Broadcast Address ("All-Ones Address")

192.0.2.255/24 → 11000000 00000000 00000010 **11111111**

What is the broadcast address of the network 192.0.2.0/23?

192.0.3.255 (11000000 00000000 00000011 11111111)

IPv4 | Allocation

IP addresses need to be unique in the network

IP address space managed by **Regional Internet Registries (RIRs)**

- **IANA** (Internet Assigned Numbers Authority)
  - **ARIN** (American Registry for Internet Numbers)
  - **LACNIC** (Latin America and Caribbean Network Information Centre)
  - **RIPE NCC** (Réseaux IP Européens Network Coordination Centre)
  - **AFRINIC** (African Network Information Center)
  - **APNIC** (Asia Pacific Network Information Centre)

⚠ IPv4 address space is considered fully exhausted since 2015

IPv4 | Special Use Addresses (RFC5735)

Range	Scope	Use
0.0.0.0/32		Unspecified / "Any Address"
127.0.0.0/8	Host	Loopback Address (the local host)
224.0.0.0/4	Global	Multicast (RFC5771)
10.0.0.0/8 172.16.0.0/12 192.168.0.0/16	Site	Private / Non-Unique Addresses (RFC1918)
169.254.0.0/16	Link	Dynamic IP Auto Configuration (RFC3927)
255.255.255.255/32	Link	Limited Broadcast

Network Address Translation (NAT)

NAT | Principle

Remap one IP address space into another

Routers do this by

- changing **source address** of an IP datagram
- changing **destination address** of an IP datagram
- changing **source port** of a TCP segment / UDP datagram
- changing **destination port** of a TCP segment / UDP datagram
- all of the above

NAT | Types of NAT

One-to-One

Map each IP address from one network to **a corresponding IP address** of another

- Merge formerly independent networks (e.g. company takeover)
- Remap public addresses without changing internal address schema (e.g. when switching ISPs)

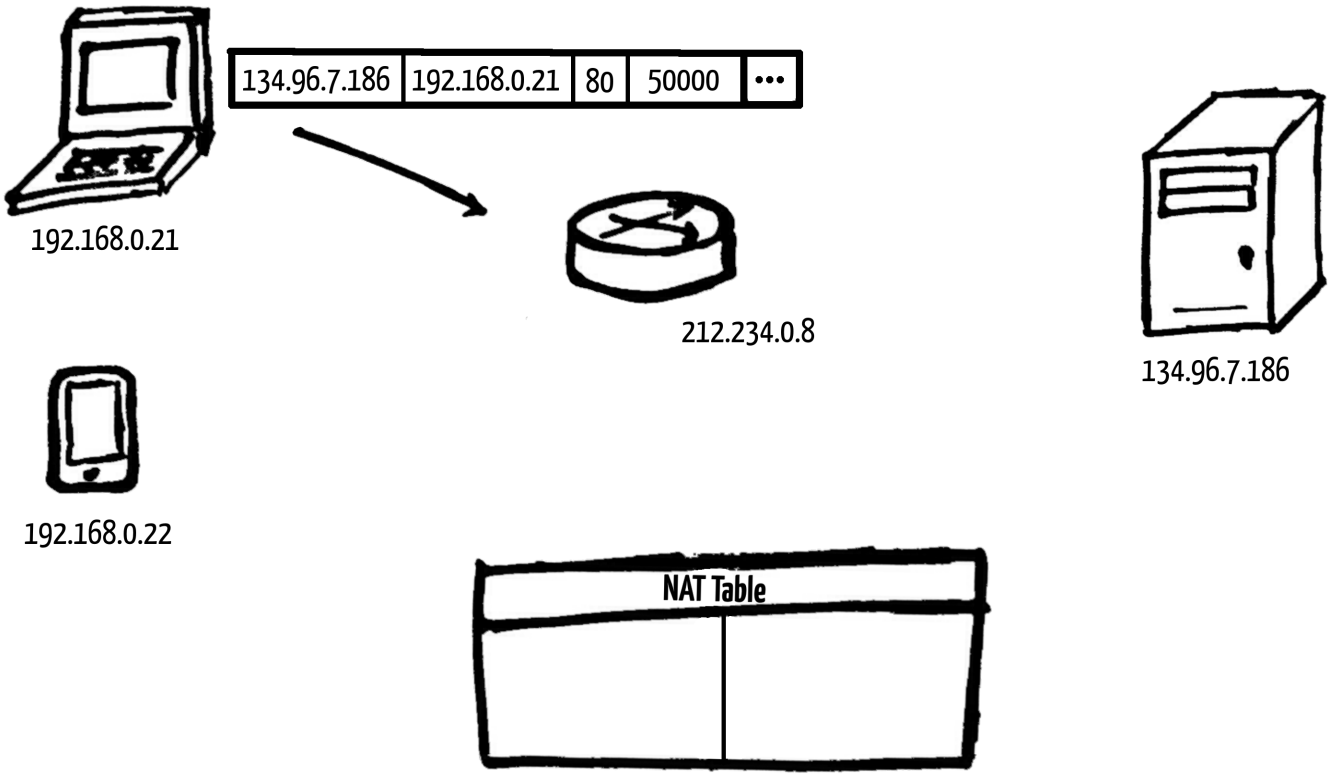
One-to-Many (Masquerading)

Map each IP address from one network to **one IP address** of another

- Connect private networks (RFC1918) to the Internet
- Conserve IP addresses

This is what people usually talk about when saying "NAT".

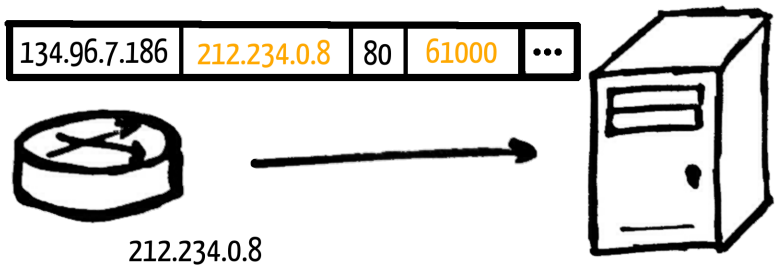
NAT | Example


134.96.7.186
192.168.0.21
80
50000
•••
192.168.0.21
212.234.0.8
134.96.7.186
192.168.0.22
NAT Table

NAT | Example

192.168.0.21

192.168.0.22

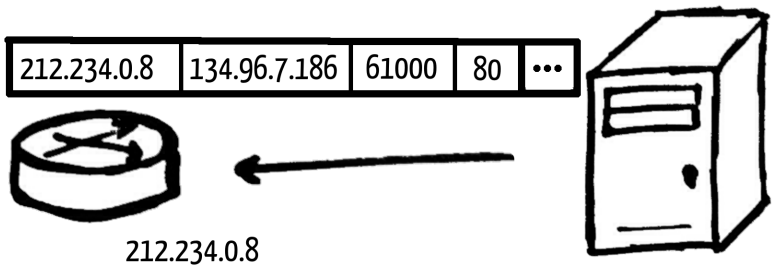


NAT Table	
192.168.0.21:50000	212.234.0.8:61000

NAT | Example

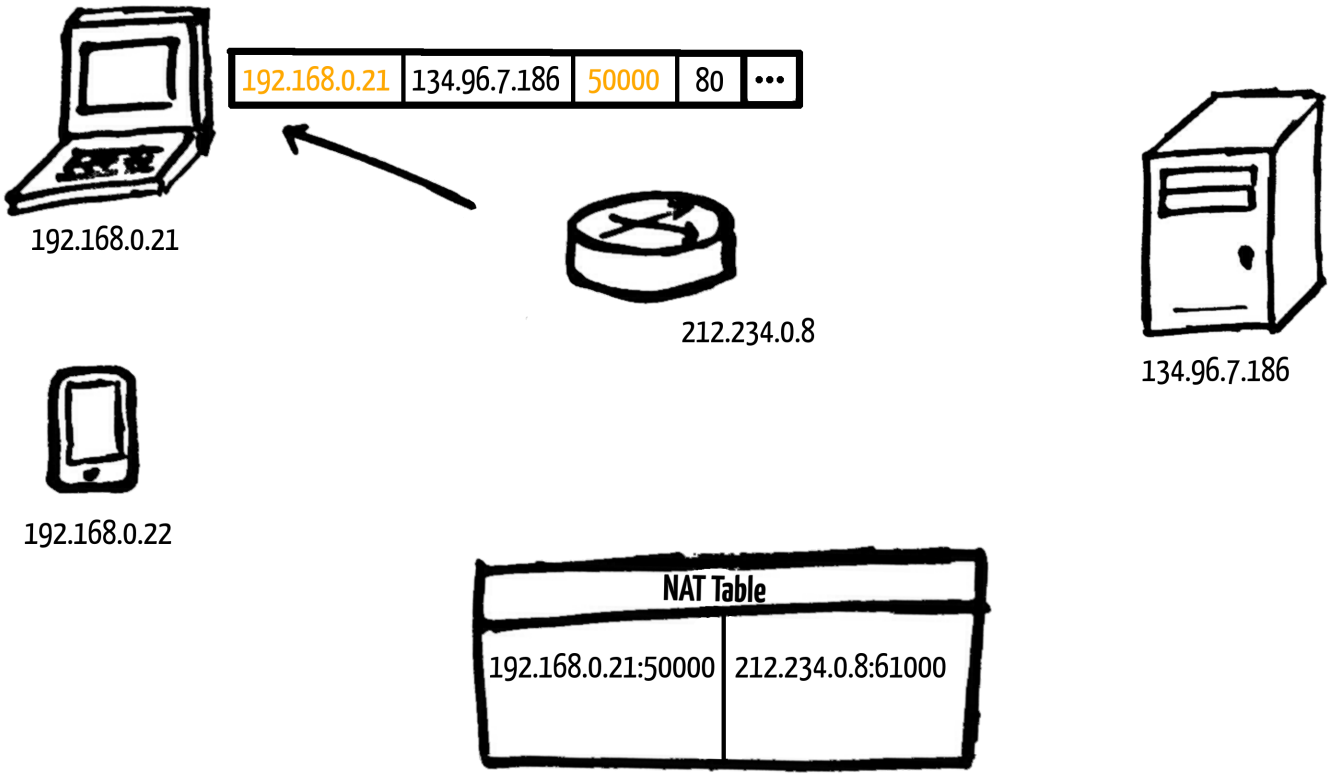
192.168.0.21

192.168.0.22

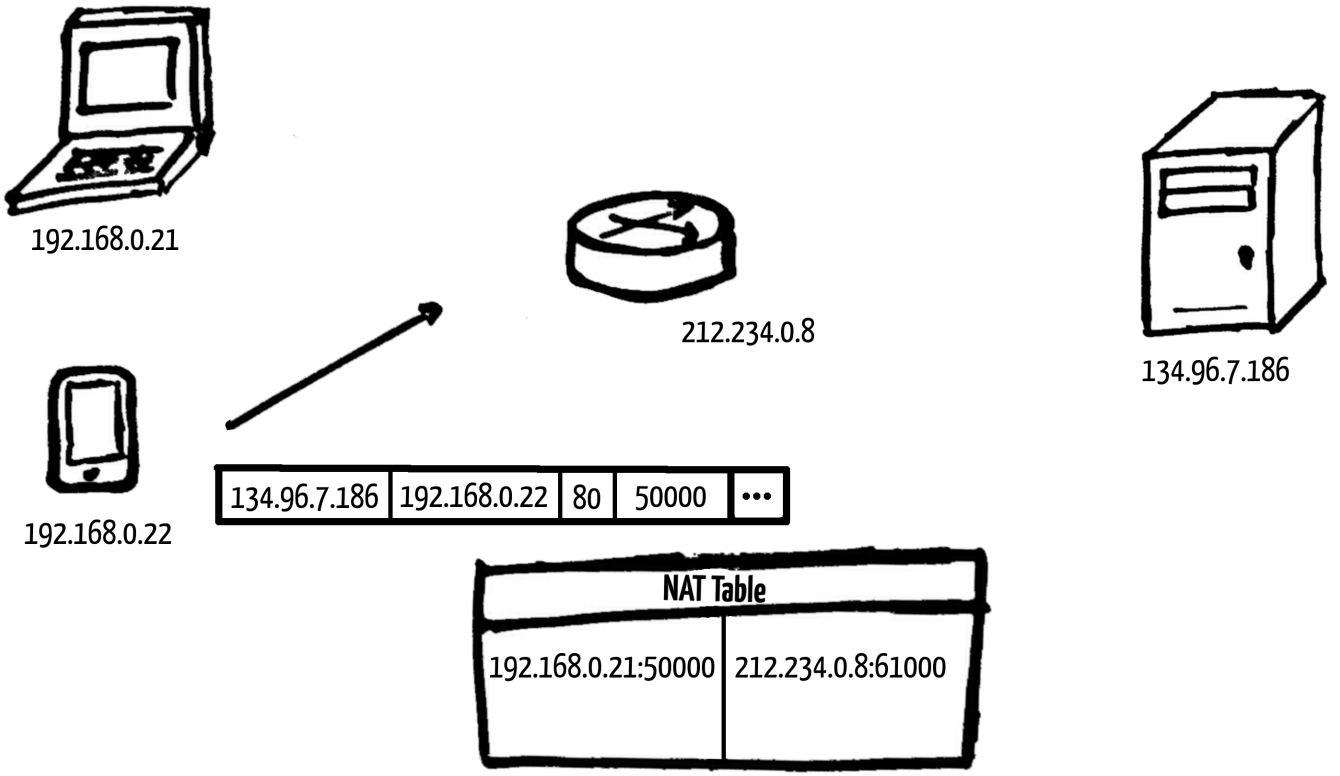


NAT Table	
192.168.0.21:50000	212.234.0.8:61000

NAT | Example


192.168.0.21
192.168.0.21
134.96.7.186
50000
80
•••
212.234.0.8
134.96.7.186
192.168.0.22
NAT Table
192.168.0.21:50000
212.234.0.8:61000

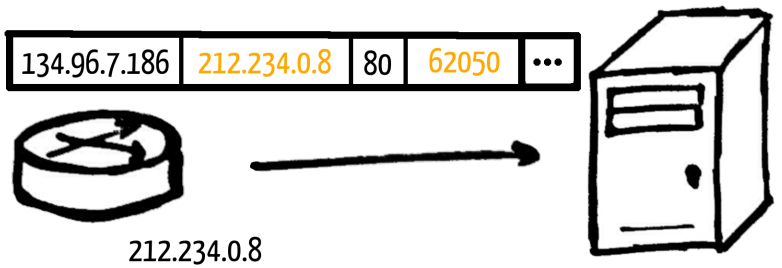
NAT | Example


192.168.0.21
212.234.0.8
134.96.7.186
192.168.0.22
134.96.7.186
192.168.0.22
80
50000
NAT Table
192.168.0.21:50000
212.234.0.8:61000

NAT | Example

192.168.0.21

192.168.0.22

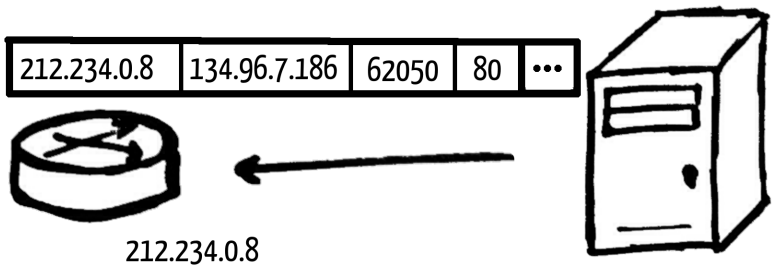


NAT Table	
192.168.0.21:50000	212.234.0.8:61000
192.168.0.22:50000	212.234.0.8:62050

NAT | Example

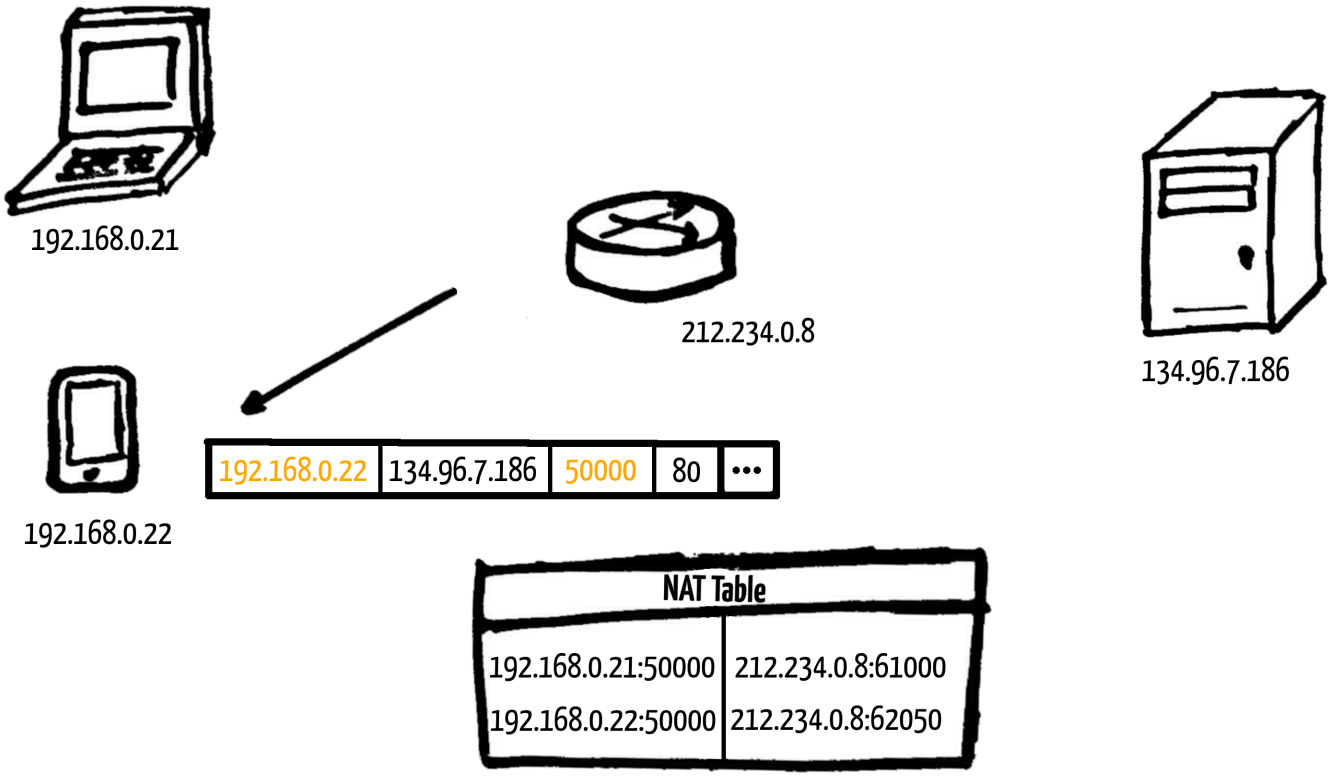
192.168.0.21

192.168.0.22



NAT Table	
192.168.0.21:50000	212.234.0.8:61000
192.168.0.22:50000	212.234.0.8:62050

NAT | Example


192.168.0.21
212.234.0.8
134.96.7.186
192.168.0.22
192.168.0.22
134.96.7.186
50000
80
•••
NAT Table
192.168.0.21:50000
212.234.0.8:61000
192.168.0.22:50000
212.234.0.8:62050

NAT

The Good

- Helps conserve IP addresses (only **one globally-scoped address**, not a subnet)
- Change ISPs without changing internal network addresses / structure
- Change internal address structure without informing outside world
- Conceals network internals from the outside world
  It is not obvious from where in the internal network a request originates
- **Internal devices not directly addressable** from the outside

NAT

The Bad

- **Internal devices not directly addressable** from the outside
- Additional **processing delay** (Checksums, Packet Reassembly)
- Can make **debugging difficult** (e.g. if there are multiple NAT routers on a path)
- Fate Sharing
  All users behind a NAT router share the same address
- Breaks the **end-to-end principle** IP was designed with
- Violates **separation between network and transport layer**
  Ports suddenly represent devices and not processes

NAT

The Ugly

- **Breaks protocols** which are unaware of or do not expect NAT
  - FTP (two connections on different ports, one for control, one for data)
  - VoIP
  - Nearly every P2P application

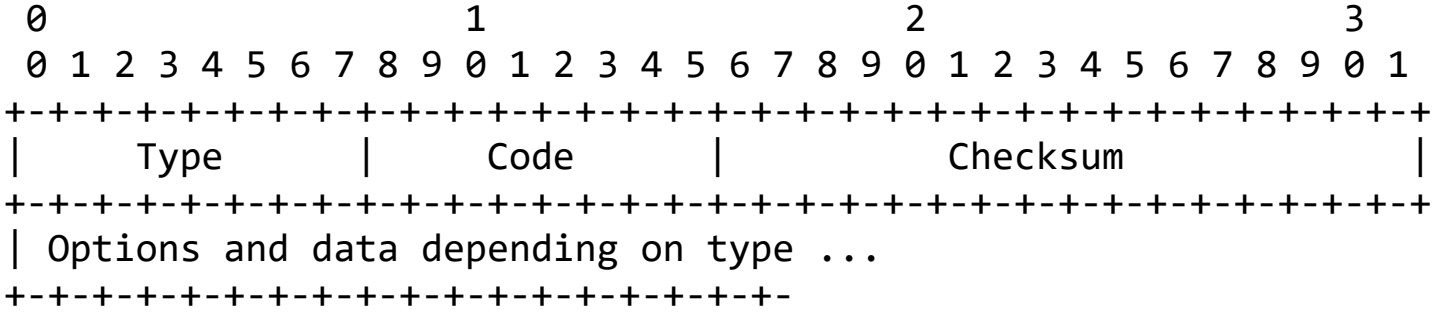
Since **NAT is very common** for home / SOHO networks, recent applications employ **NAT Traversal** techniques to establish connections between NATed hosts.

Internet Control Message Protocol

ICMP (Version 4)

- Defined in RFC792
- IP protocol number: **1**
- Used to **send/receive error and diagnostic messages** in IPv4 networks
- **Error messages** contain the **IP header and the first 8 bytes** of the **packet that caused the error**

Technically ICMP is a transport layer protocol but is considered to be part of the network layer



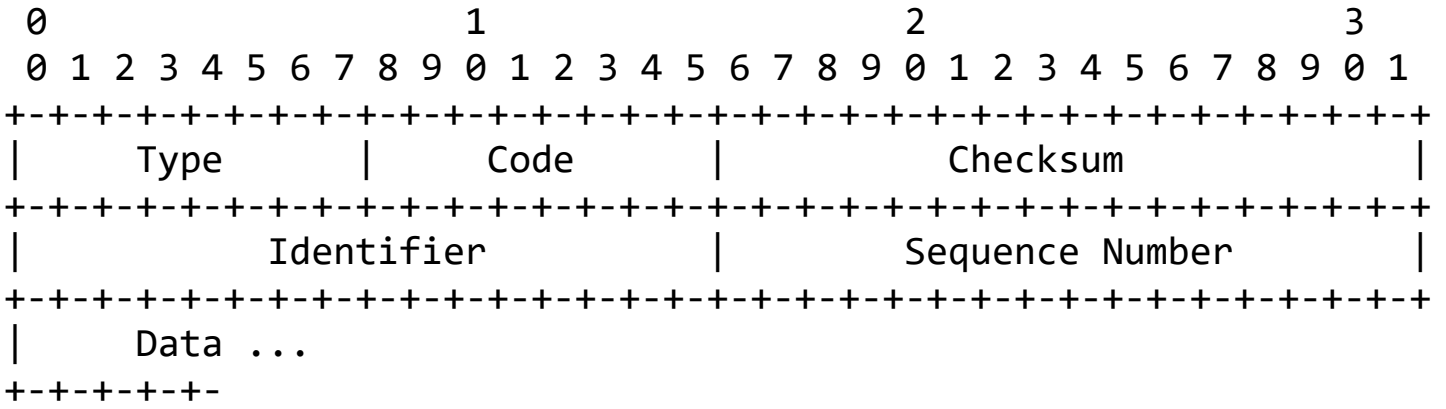
ICMPv4 | Common Types and Codes

Type	Code	Description
0	0	ECHO Reply
3		**Destination Unreachable**
	1	
	3	
	4	
5		**Redirect Message**
	0	
	1	
8	0	ECHO Request
9	0	Router Advertisement
10	0	Router Solicitation
11	0	TTL Exceeded

ICMPv4 | Ping

Check host connectivity by sending an ICMP echo and waiting for a corresponding reply

Type	Code	Description
0	0	ECHO Reply
8	0	ECHO Request



ICMPv4 | Ping

- ping (, ,)
- nmap (; ,)

```
$ ping -c 1 134.96.7.186
PING 134.96.7.186 (134.96.7.186) 56(84) bytes of data.
64 bytes from 134.96.7.186: icmp_seq=1 ttl=63 time=0.290 ms

--- 134.96.7.186 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2998ms
rtt min/avg/max/mdev = 0.287/0.317/0.355/0.029 ms
```

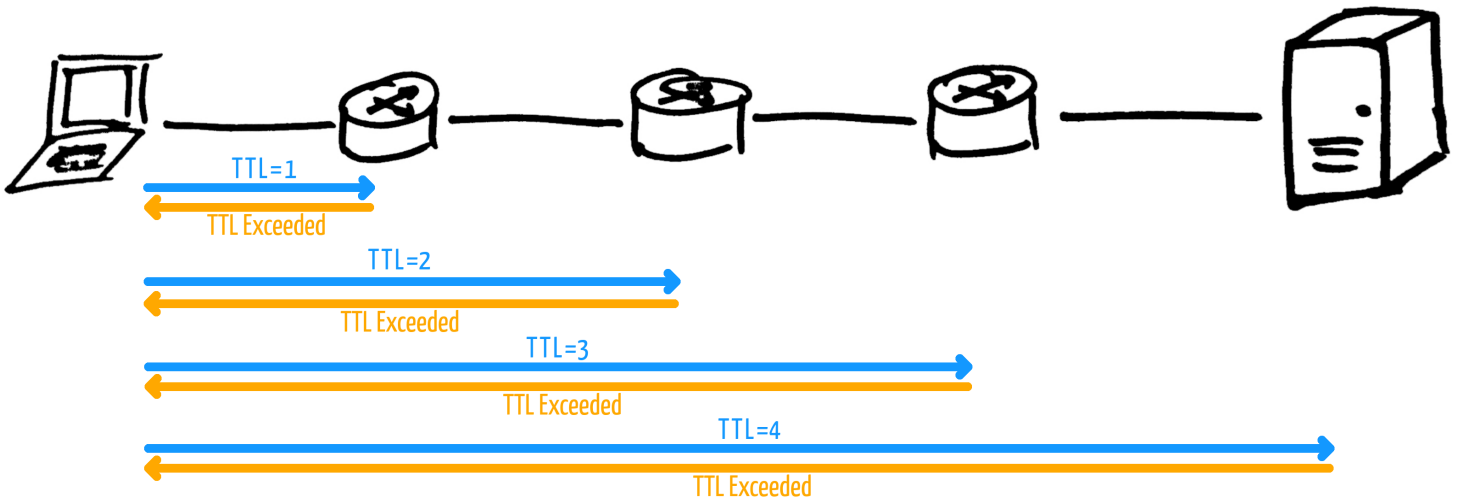
```
$ nmap -sP 134.96.7.186-187

Starting Nmap 6.40 ( http://nmap.org ) at 2016-08-23 16:38 CEST
Nmap scan report for web-f6.rz.uni-saarland.de (134.96.7.186)
Host is up (0.00040s latency).
Nmap scan report for web-f7.rz.uni-saarland.de (134.96.7.187)
Host is up (0.00038s latency).
Nmap done: 2 IP addresses (2 hosts up) scanned in 0.00 seconds
```

ICMPv4 | Path Tracing

Goal: Find the path over which a datagram travels to its destination.

Make clever use of datagram TTL.



ICMPv4 | Path Tracing

- traceroute (, ,)
- tracepath ()

```
$ sudo traceroute -I 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  vlan-herfet-neu.nt.uni-saarland.de (134.96.86.1)  0.439 ms  0.409 ms  0.410 ms
 2  c65eb36-win.net.uni-saarland.de (134.96.6.54)  0.343 ms  0.379 ms  0.385 ms
 3  xr-saa1-pc1.x-win.dfn.de (188.1.234.37)  0.940 ms  1.061 ms  1.245 ms
 4  cr-dui1-te0-7-0-3.x-win.dfn.de (188.1.146.85)  8.237 ms  8.337 ms  8.344 ms
 5  cr-fra2-hundredgige0-9-0-3.x-win.dfn.de (188.1.144.178)  12.520 ms  12.584 ms  12.677 ms
 6  72.14.194.58 (72.14.194.58)  12.658 ms  12.337 ms  12.439 ms
 7  216.239.47.84 (216.239.47.84)  12.380 ms  12.372 ms  12.365 ms
 8  216.239.57.147 (216.239.57.147)  34.235 ms  33.448 ms  33.410 ms
 9  66.249.95.23 (66.249.95.23)  15.488 ms  15.486 ms  15.565 ms
10  216.58.215.253 (216.58.215.253)  19.929 ms  20.163 ms  20.127 ms
11  108.170.234.47 (108.170.234.47)  20.082 ms  19.281 ms  19.259 ms
12  * * *
13  google-public-dns-a.google.com (8.8.8.8)  18.983 ms  18.953 ms  18.948 ms
```

ICMP | Exploits 

- ICMP tunneling using ECHO requests / replies
  Establish a covert channel using the data portion of ICMP Echo requests and replies. Can be difficult to detect.

- DDos Attacks using ECHO requests / replies
  Send ECHO request with a forged source IP to e.g. a network's broadcast address (e.g. Smurf Attack).

- Forged ICMP Redirects / Router Advertisements
  Enables Man-in-the-middle Attacks.

- "Ping of Death"
  Sending a malformed ICMP ECHO request to a host with a flawed network implementation.

At the Bar... again

An ICMP packet walks into a bar, says "Hello!" to the bartender, who then in turn runs out to tell the ICMP packet's wife.

Source

Wrap-Up

Questions?

Take-Home Messages

- The network layer provides **host-to-host connectivity across link boundaries**.
- Routing and forwarding are conceptually **two different things**.
- **Datagram networks** are on the rise for arbitrary data.
- **Virtual circuit networks** still exist, but are **losing importance**.
- **Network Address Translation (NAT)** is a common occurence in the IPv4 Internet, but **has its flaws**.
- **ICMP** facilitates network control and debugging in **IPv4 networks**.

Further Reading

- Kurose-Ross "Computer Networking" (Sections 4.1, 4.2, 4.4)
- All mentioned RFCs