# Network Basics

Wang Xiaolin
wx672ster+net@gmail.com

December 12, 2019

# Textbooks

- TANENBAUM A, WETHERALL D. *Computer Networks*. Pearson Prentice Hall, 2011.
- KUROSE J, ROSS K. *Computer Networking: A Top-down Approach*. Pearson, 2013.
- FALL K, STEVENS W. *TCP/IP Illustrated, Volume 1: The Protocols*. Pearson Education, 2011.

# Course Web Links

Course web site https://cs6.swfu.edu.cn/moodle

Lecture slides https://cs6.swfu.edu.cn/~wx672/lecture_notes/network_basics/

Lab exercises https://cs6.swfu.edu.cn/~wx672/lecture_notes/network_basics/net-tools/net-tools.html

Cisco Networking Academy https://www.netacad.com/

Beej’s Guides http://beej.us/guide/

# Homework

## Weekly tech question

1. What was I trying to do?
2. How did I do it? (steps)
3. The expected output? The real output?
4. How did I try to solve it? (steps, books, web links)
5. How many hours did I struggle on it?

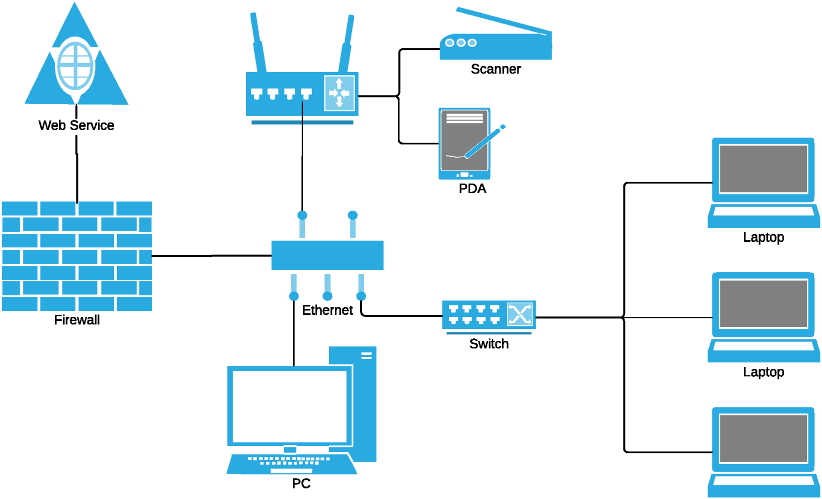
✉ `wx672ster+linux@gmail.com`

𝔼 Preferably in English

in stackoverflow style

OR simply show me the tech questions you asked on any website

# What's A Computer Network?



# The History of Internet I

1836: Telegraph

1858-66: Transatlantic cable

1876: Telephone

1957: USSR launches Sputnik

1962-68: *Packet-switching* networks developed

1969: Birth of Internet

1971: People communicate over a network

1972: Computers can connect more freely and easily

1973: Global Networking becomes a reality

1974: Packets become mode of transfer

1976: Networking comes to many

# The History of Internet II

1977: E-mail takes off, Internet becomes a reality

1979: News Groups born

1981: Things start to come together

1982: *TCP/IP* defines future communication

1983: Internet gets bigger

1984: Growth of Internet Continues

1986: Power of Internet Realised

1987: Commercialisation of Internet Born

1989: Large growth in Internet

1990: Expansion of Internet continues

1991: Modernisation Begins

# The History of Internet III

1992: Multimedia changes the face of the Internet

1993: The WWW Revolution truly begins

1994: Commercialisation begins

1995: Commercialisation continues apace

1996: Microsoft enters

1998: Google

Homework: Meanwhile, what happened in China?

# What’s The Internet?

What pops up in your mind if I say “Internet”?

# What’s The Internet?

What pops up in your mind if I say “Internet”?

For me, the answer is...

Google

and...

# What’s The Internet?

What pops up in your mind if I say “Internet”?

For me, the answer is...

Google

and...

TCP/IP

# What’s The Internet?

- The network of networks.
- Tech view: `TCP/IP`
- App view: Google

# Google Philosophy

Ten things Google has found to be true

1. Focus on the user and all else will follow.
2. It’s best to do one thing really, really well.
3. Fast is better than slow.
4. Democracy on the web works.
5. You don’t need to be at your desk to need an answer.
6. *You can make money without doing evil.*
7. There’s always more information out there.
8. The need for information crosses all borders.
9. You can be serious without a suit.
10. Great just isn’t good enough.

# Google Philosophy

## More about...

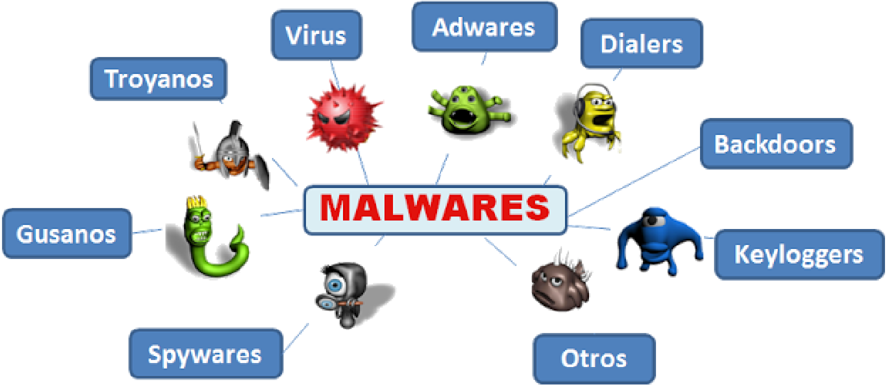
- Software Principles
- Google User Experience
- No pop-ups
- Security

# Google Products

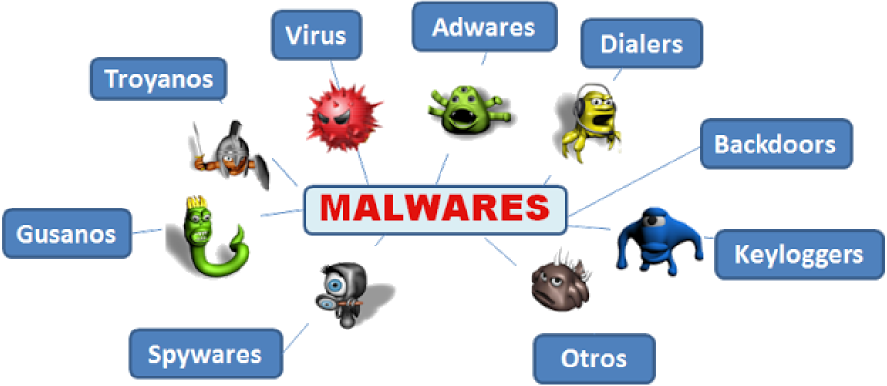
# Choosing The Right Tools

vs.

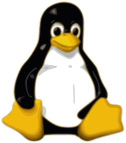
# Dangerous


Troyanos
Virus
Adwares
Dialers
Backdoors
MALWARES
Gusanos
Keyloggers
Spywares
Otros

# Dangerous



# My solution



# Safe Surfing Advice

## Take care of your identity and privacy

- Use a better browser, and keep it updated
- Use a spam filter for emailing
- Always use strong passwords
- Don’t give away too much personal information on blogs and social networking sites

# Safe Surfing Advice

## Protect Your PC

- Get anti-virus software, anti-spyware software and a firewall
- Keep your computer up to date
- Block spam emails
- Use an up to date web browser
- Make regular backups
- Encrypt your wireless network

# Safe Surfing Advice

## Avoid online rip-offs

- When you’re shopping online, look for clear signs that you’re buying from a reputable company
- On an online auction site, learn how it works and learn to pick good sellers
- Use safe ways to pay, such as PayPal or credit and debit cards
- Use your common sense to avoid scams –if it sounds too good to be true, it probably is

# Homework

1. try 🐧
2. get a gmail account
3. recommend a good chrome extension to me via gmail
4. in google plus, share an interesting post to me
5. add your class timetable into google calendar, and then share your calendar to me
6. in youtube, find a video you like and share it to me

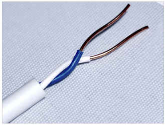
# Network Classification

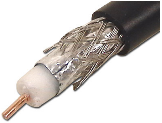
- connection method: wired, wireless...
- topology
- scale
- network architecture: c/s, p2p...

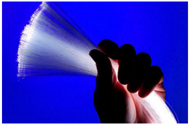
# Network Classification

## Connection method

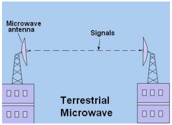
Wired:

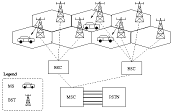






Wireless:



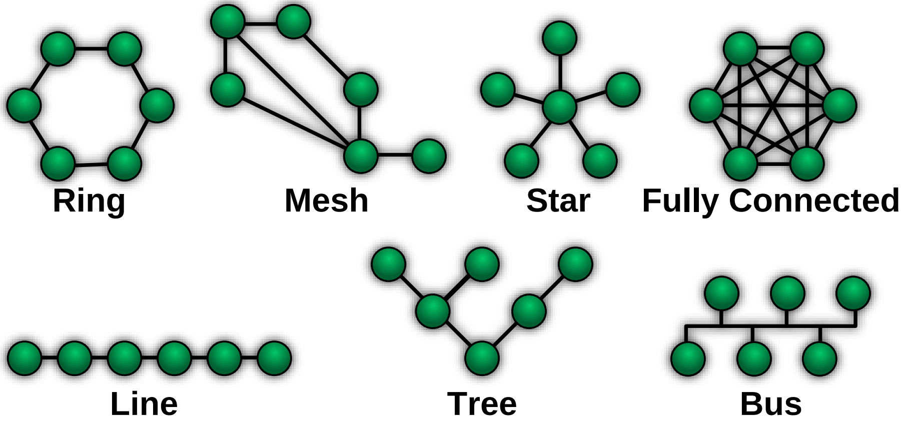


CONNECTED
CONNECTED

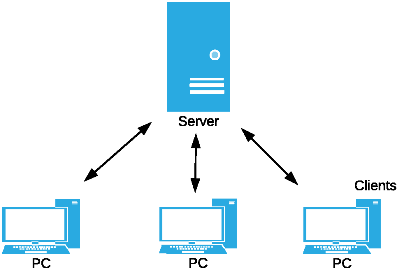
## Scale

PAN, LAN, CAN, MAN, WAN ...

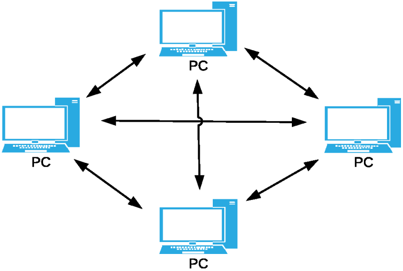
## Topology



# Network Architecture



Client/Server

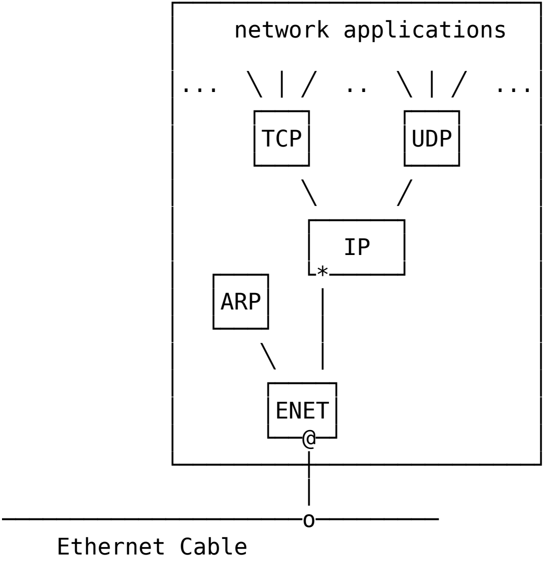


Peer to Peer

# Basic Hardware Components

| | | | | | | |
|---|---|---|---|---|---|---|
| IP | | | | | Router: | |
| Link | | | Bridge: | | Switch: | |
| PHY | NIC: | | Repeater: | | Hub: | |

# TCP/IP



network applications

... .. ...

TCP

UDP

IP

ARP

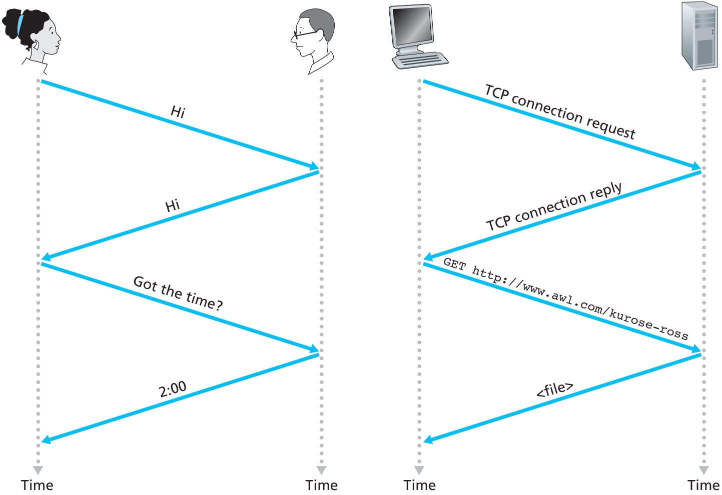
ENET

Ethernet Cable

# What's TCP/IP?

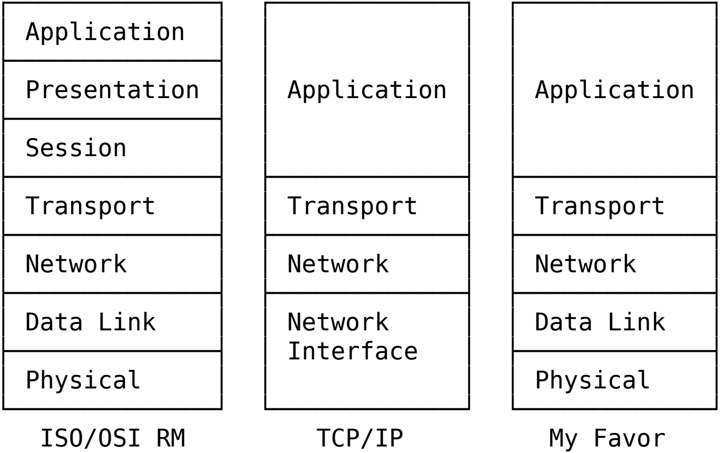
TCP/IP A set of protocols designed for the Internet

protocol a rule, a treaty, an agreement ...



# TCP/IP Protocol Stack

## Every networked computer has it inside



Application

Presentation

Session

Transport

Network

Data Link

Physical

ISO/OSI RM

Application

Transport

Network

Network Interface

TCP/IP

Application

Transport

Network

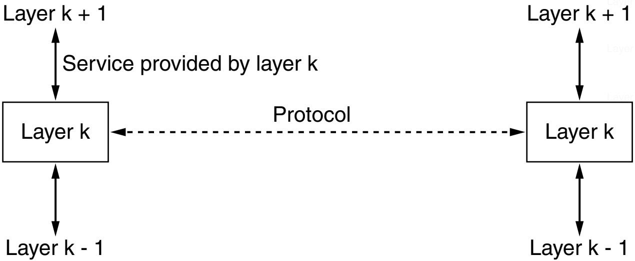
Data Link

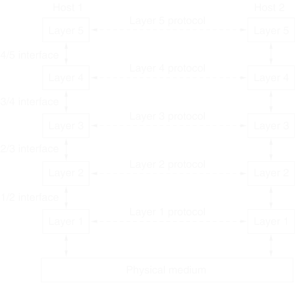
Physical

My Favor

# Layered Design

## Services vs. Protocols

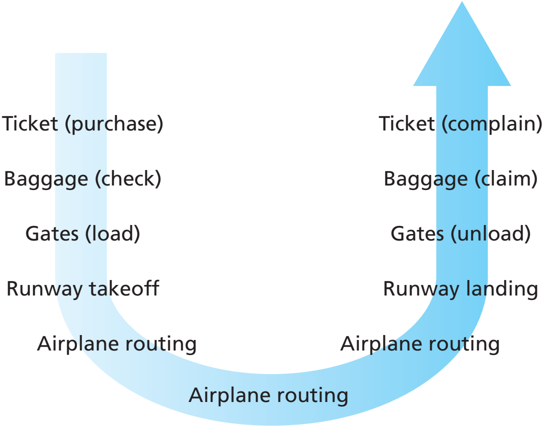




| **Services** | **Protocols** |
|---|---|
| Layer to Layer | Peer to Peer |
| A set of operations<br>`(listen, connect, accept, receive, send, disconnect)` | A set of rules<br>`(message format, message meanings)` |

# Layered Design Example

## Taking an airplane trip



Ticket (purchase)

Baggage (check)

Gates (load)

Runway takeoff

Airplane routing

Airplane routing

Airplane routing

Runway landing

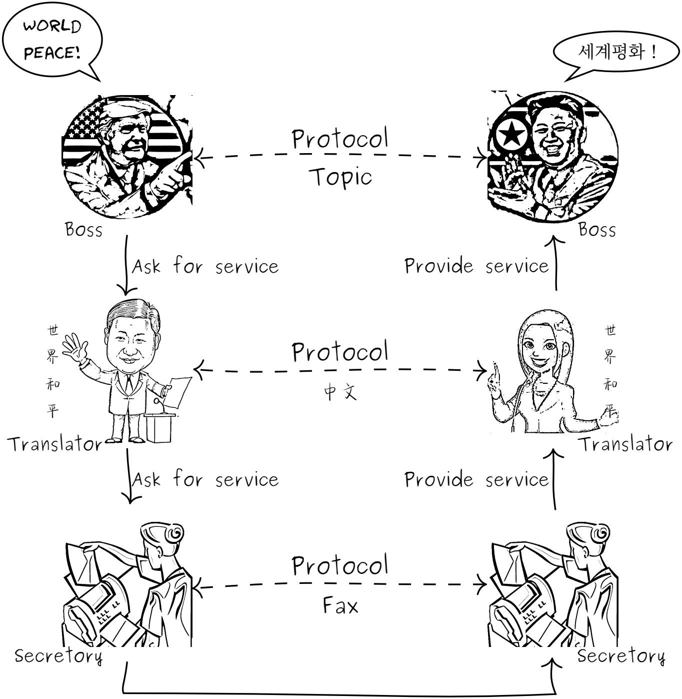
Gates (unload)

Baggage (claim)

Ticket (complain)

Each layer

1. has some functions
2. provides services to its upper layer



Boss

Protocol

Topic

Boss

Ask for service

Provide service

Translator

Protocol

中文

Translator

Ask for service

Provide service

Secretory

Protocol

Fax

Secretory

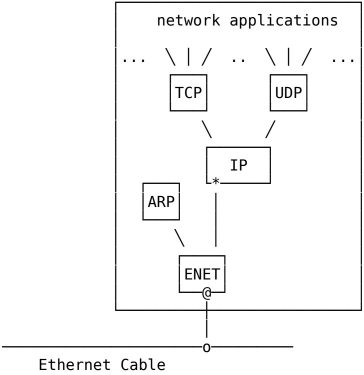
## Each protocol is completely independent of the other ones

For example

- The translators (L2) can switch from Dutch to Finnish without touching L1 or L3
- The secretaries (L1) can switch from email to telephone without disturbing (or even informing) the other layers

# TCP/IP Overview

## Basic Structure



1. Where will an incoming Ethernet frame go?

   0x0806 → ARP
   0x0800 → IP

2. Where will an incoming IP packet go?

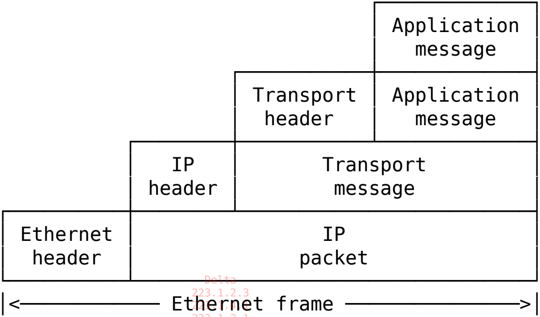
   0x06 → TCP
   0x11 → UDP

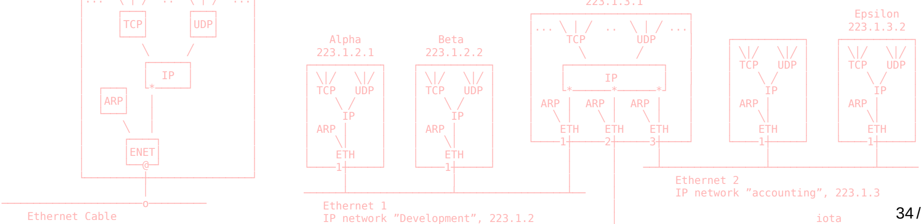
3. Where will an incoming transport message (UDP datagram, TCP segment) go?

| HTTP | FTP | SSH | SMTP |
|---|---|---|---|
| 80 | 21/20 | 22 | 25 |

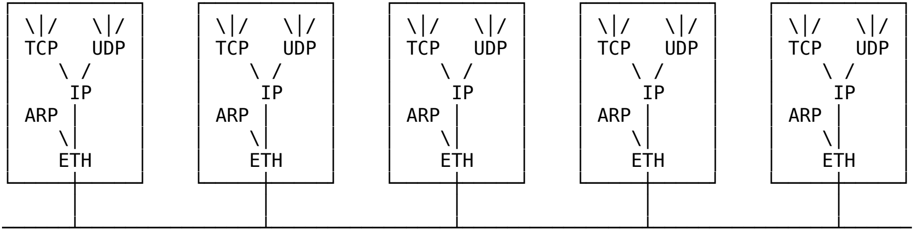
# The Name Of A Unit Of Data

- A Application message
- T TCP segment; UDP datagram
- N IP packet
- L Ethernet frame





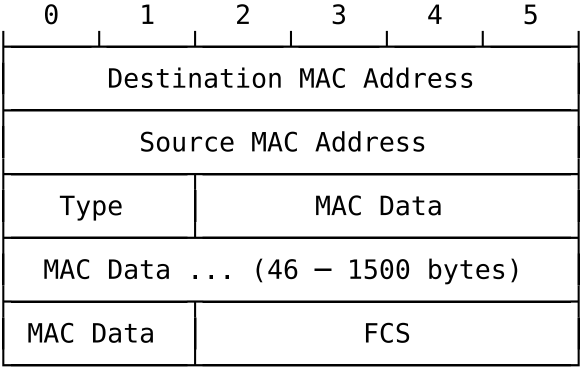
# Ethernet



```
 ____________     ____________     ____________     ____________     ____________
|            |   |            |   |            |   |            |   |            |
| \|/   \|/  |   | \|/   \|/  |   | \|/   \|/  |   | \|/   \|/  |   | \|/   \|/  |
| TCP   UDP  |   | TCP   UDP  |   | TCP   UDP  |   | TCP   UDP  |   | TCP   UDP  |
|    \ /     |   |    \ /     |   |    \ /     |   |    \ /     |   |    \ /     |
|     IP     |   |     IP     |   |     IP     |   |     IP     |   |     IP     |
| ARP |      |   | ARP |      |   | ARP |      |   | ARP |      |   | ARP |      |
|    \|      |   |    \|      |   |    \|      |   |    \|      |   |    \|      |
|    ETH     |   |    ETH     |   |    ETH     |   |    ETH     |   |    ETH     |
|_____|______|   |_____|______|   |_____|______|   |_____|______|   |_____|______|
      |                |                |                |                |
______|________________|________________|________________|________________|_______
```

1. Frame format?
2. Address format?
3. Broadcast address?
4. CSMA/CD?

# Ethernet Frame



0 1 2 3 4 5

Destination MAC Address

Source MAC Address

Type

MAC Data

MAC Data ... (46 – 1500 bytes)

MAC Data

FCS

# Examples

## Unicast, carrying an IP packet

| Destination | Source | Type | MAC Data | |
|---|---|---|---|---|
| 08005A21A722 | 0800280038A9 | 0x0800 | IP packet | FCS |

## Unicast, carrying an ARP packet

| Destination | Source | Type | MAC Data | |
|---|---|---|---|---|
| 0800280038A9 | 08005A21A722 | 0x0806 | ARP Response | FCS |

## Broadcast, carrying an ARP packet

| Destination | Source | Type | MAC Data | |
|---|---|---|---|---|
| FFFFFFFFFFFF | 0800280038A9 | 0x0806 | ARP Request | FCS |

# Ethernet References

- Wikipedia. *Ethernet — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Ethernet&oldid=648082976.

- Wikipedia. *Ethernet frame — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Ethernet%5C_frame&oldid=653337888.

- Wikipedia. *Carrier sense multiple access with collision detection — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Carrier%5C_sense%5C_multiple%5C_access%5C_with%5C_collision%5C_detection&oldid=648771859.

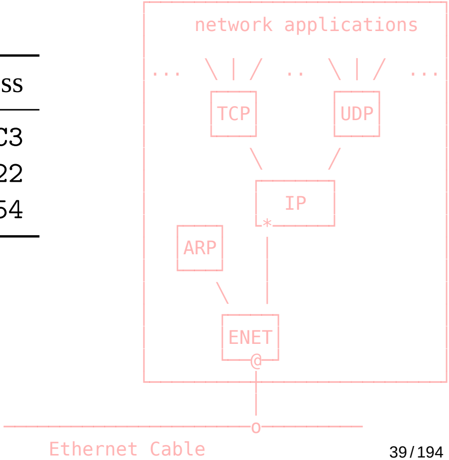
- POSTEL J, REYNOLDS J. *Standard for the transmission of IP datagrams over IEEE 802 networks*. RFC Editor. 1988. https://www.rfc-editor.org/rfc/rfc1042.txt.

# ARP

ARP Looking up the ARP table to find the destination MAC address.

## Example ARP table

| IP address | Ethernet address |
|---|---|
| 223.1.2.1 | 08-00-39-00-2F-C3 |
| 223.1.2.3 | 08-00-5A-21-A7-22 |
| 223.1.2.4 | 08-00-10-99-AC-54 |



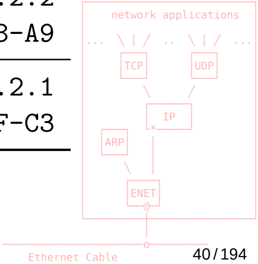
# Where does the ARP table come from?

## Example ARP Request

| | |
|---|---|
| Sender IP Address | 223.1.2.1 |
| Sender Eth Address | 08-00-39-00-2F-C3 |
| Target IP Address | 223.1.2.2 |
| Target Eth Address | FF-FF-FF-FF-FF-FF |

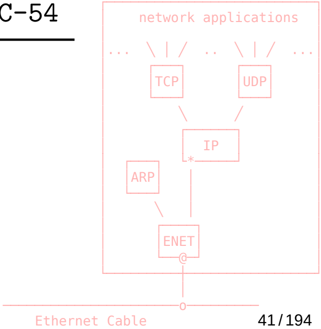
## Example ARP Response

| | |
|---|---|
| Sender IP Address | 223.1.2.2 |
| Sender Eth Address | 08-00-28-00-38-A9 |
| Target IP Address | 223.1.2.1 |
| Target Eth Address | 08-00-39-00-2F-C3 |



## The updated table

| IP address | Ethernet address |
|---|---|
| 223.1.2.1 | 08-00-39-00-2F-C3 |
| 223.1.2.2 | 08-00-28-00-38-A9 |
| 223.1.2.3 | 08-00-5A-21-A7-22 |
| 223.1.2.4 | 08-00-10-99-AC-54 |



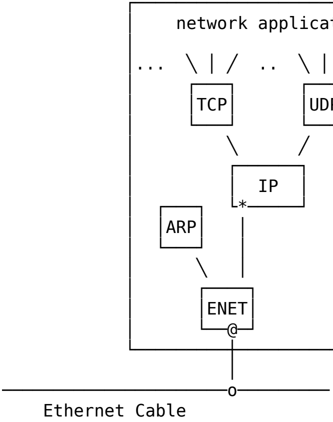
# ARP References

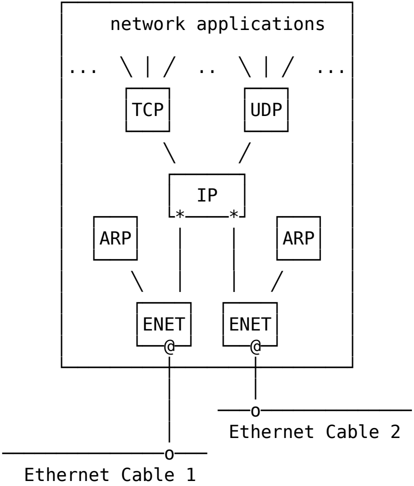
- Wikipedia. *Address Resolution Protocol — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Address%5C_Resolution%5C_Protocol&oldid=647148843.

- PLUMMER D. *An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware*. RFC Editor. 1982. https://www.rfc-editor.org/rfc/rfc826.txt.

# IP

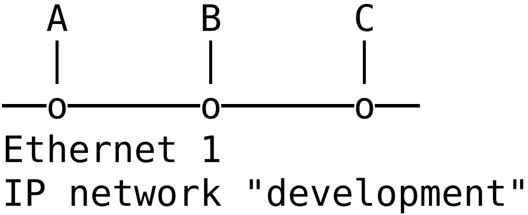
## Router





**Routing** Find a route in the route table.

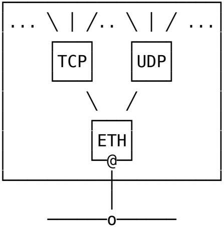
# Direct Routing—IP is overhead

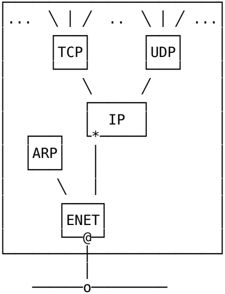


Addresses in an Ethernet frame for an IP packet from A to B

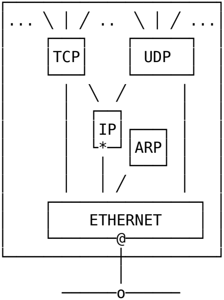
| address | source | destination |
|---|---|---|
| IP | A | B |
| Eth | A | B |

# Is IP Necessary?

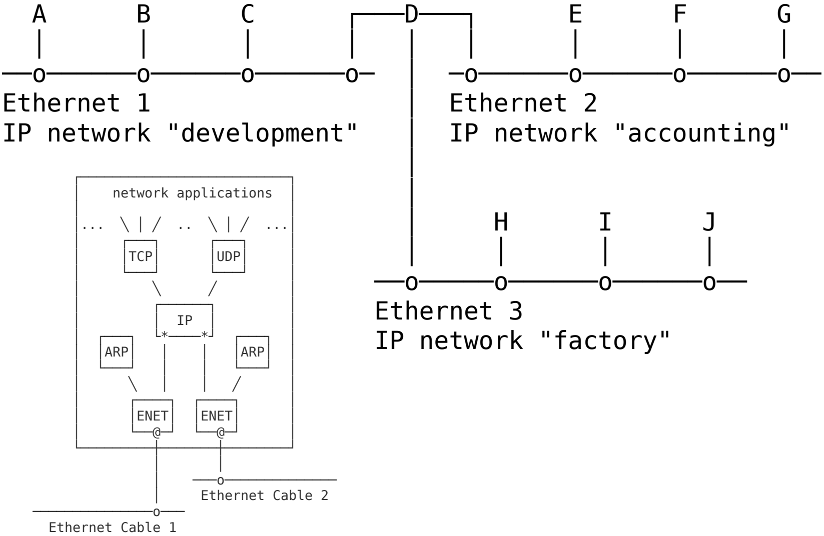




Dual stack



# Indirect Routing



A B C D E F G

Ethernet 1
IP network "development"

Ethernet 2
IP network "accounting"

H I J

Ethernet 3
IP network "factory"

network applications

... .. ...

TCP UDP

IP

ARP ARP

ENET ENET

Ethernet Cable 2

Ethernet Cable 1

## Addresses in an Ethernet frame for an IP packet from A to E (before D)

| address | source | destination |
|---|---|---|
| IP | A | E |
| Eth | A | D |

## Addresses in an Ethernet frame for an IP packet from A to E (after D)

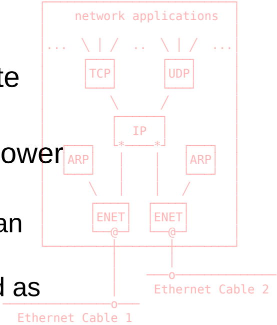
| address | source | destination |
|---|---|---|
| IP | A | E |
| Eth | D | E |

```
  A      B      C     ┌──D──┐     E      F      G
  |      |      |     |  |  |     |      |      |
──o──────o──────o─────o─ | ─o─────o──────o──────o──
Ethernet 1               |  Ethernet 2
IP network "development" |  IP network "accounting"
                         |
```

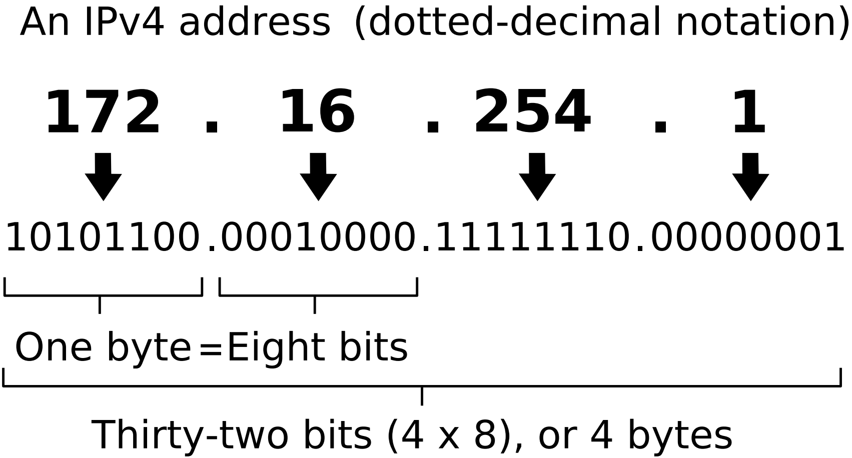
# IP Module Routing Rules

1. For an outgoing IP packet, entering IP from an upper layer, IP must decide
   - whether to send the IP packet directly or indirectly, and
   - IP must choose a lower network interface.

   These choices are made by consulting the route table.
2. For an incoming IP packet, entering IP from a lower interface, IP must decide
   - whether to forward the IP packet or pass it to an upper layer.
   - If the IP packet is being forwarded, it is treated as an outgoing IP packet.
3. When an incoming IP packet arrives it is never forwarded back out through the same network interface.



# IP Address



An IPv4 address (dotted-decimal notation)

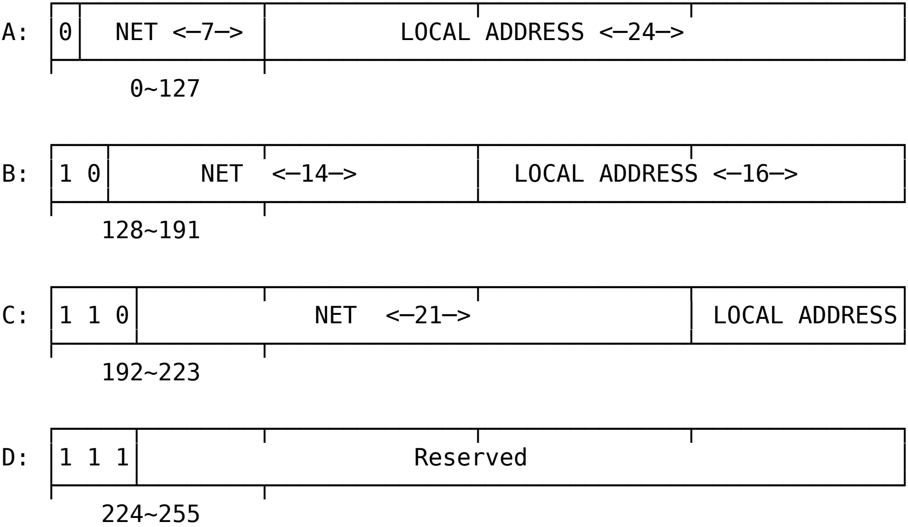
**172 . 16 . 254 . 1**

10101100.00010000.11111110.00000001

One byte = Eight bits

Thirty-two bits (4 x 8), or 4 bytes

# Address classes



A: 0 NET <–7–> LOCAL ADDRESS <–24–>

0~127

B: 1 0 NET <–14–> LOCAL ADDRESS <–16–>

128~191

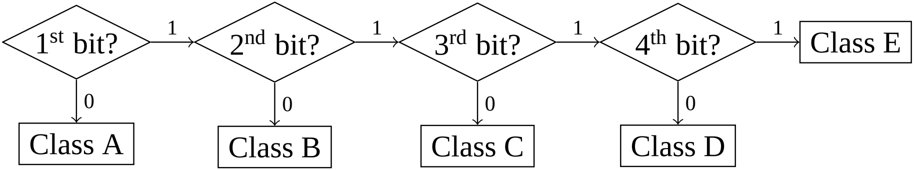
C: 1 1 0 NET <–21–> LOCAL ADDRESS

192~223

D: 1 1 1 Reserved

224~255

# Prefix


1st bit?
1
2nd bit?
1
3rd bit?
1
4th bit?
1
Class E
0
Class A
0
Class B
0
Class C
0
Class D

# Special IP Addresses

- A value of zero in the network field means this network. (source only)
- A value of zero in the host field means network address.
- `127.x.x.x` are loopback address.
- `255.255.255.255` is boardcast address.
- Private address:
  - `10.x.x.x`
  - `172.16.x.x`∼`172.31.x.x`
  - `192.168.x.x`
- CIDR—Classless Inter-Domain Routing—An IP addressing scheme that replaces the older system based on classes A, B and C.

# Names

People refer to computers by names, not numbers.

/etc/hosts

```
127.0.0.1         localhost
202.203.132.245   cs3.swfu.edu.cn   cs3
```

/etc/networks

```
localnet    202.203.132.192
```

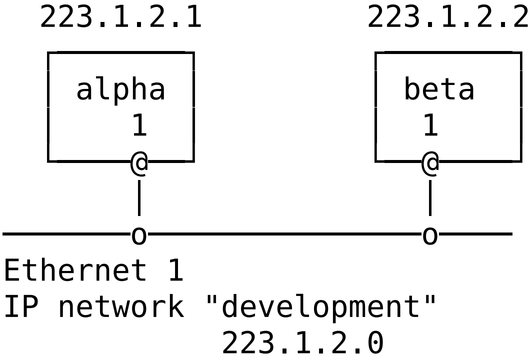
# IP Route Table

## Example IP Route Table

```
~$ route
Kernel IP routing table
Destination    Gateway          Iface
localnet       *                eth0
192.168.128.0  *                eth0
default        202.203.132.254  eth0
```

```
~$ man route
```

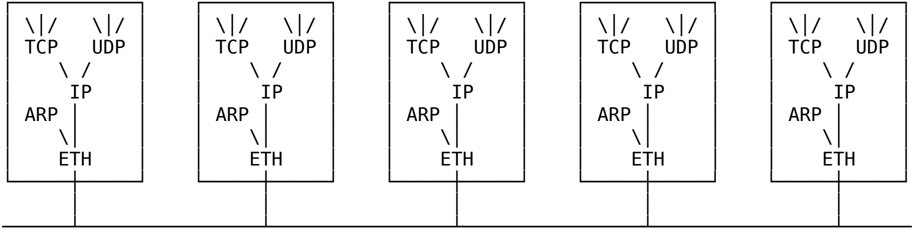
# Direct Routing Details



## The route table inside alpha (simplified)

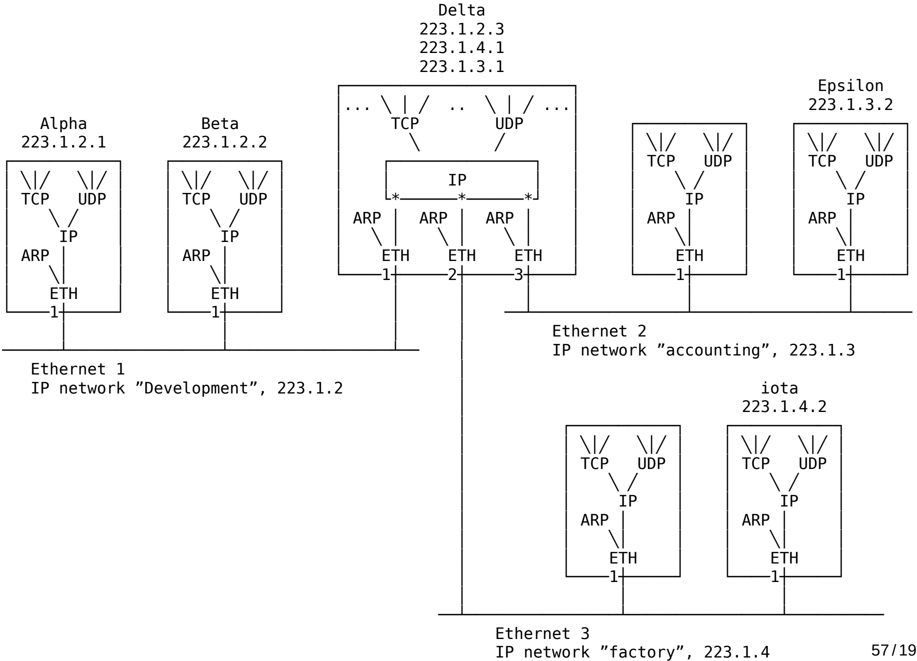
| network | flag | router | interface |
|---|---|---|---|
| development | direct | | 1 |

# Homework



Alpha is sending an IP packet to beta…Please describe.

# Indirect Routing Details



Delta
223.1.2.3
223.1.4.1
223.1.3.1

... TCP .. UDP ...
IP
ARP ARP ARP
ETH ETH ETH
1 2 3

Alpha
223.1.2.1
TCP UDP
IP
ARP
ETH
1

Beta
223.1.2.2
TCP UDP
IP
ARP
ETH
1

TCP UDP
IP
ARP
ETH
1

Epsilon
223.1.3.2
TCP UDP
IP
ARP
ETH
1

Ethernet 1
IP network ”Development”, 223.1.2

Ethernet 2
IP network ”accounting”, 223.1.3

TCP UDP
IP
ARP
ETH
1

iota
223.1.4.2
TCP UDP
IP
ARP
ETH
1

Ethernet 3
IP network ”factory”, 223.1.4

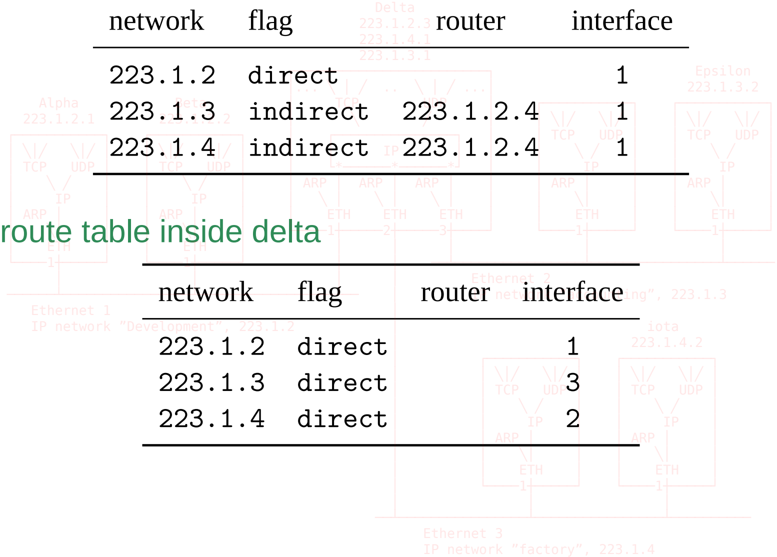
# Indirect Routing Details

## The route table inside alpha

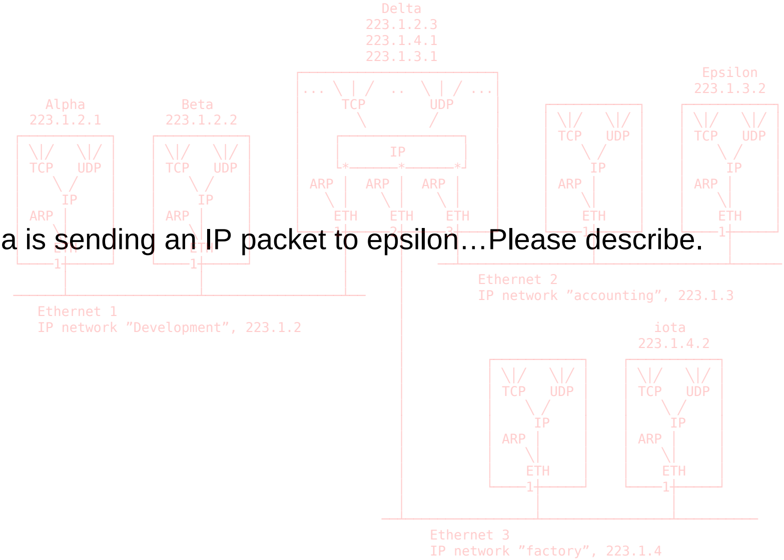
| network | flag | router | interface |
|---|---|---|---|
| 223.1.2 | direct | | 1 |
| 223.1.3 | indirect | 223.1.2.4 | 1 |
| 223.1.4 | indirect | 223.1.2.4 | 1 |

## The route table inside delta

| network | flag | router | interface |
|---|---|---|---|
| 223.1.2 | direct | | 1 |
| 223.1.3 | direct | | 3 |
| 223.1.4 | direct | | 2 |



# Homework

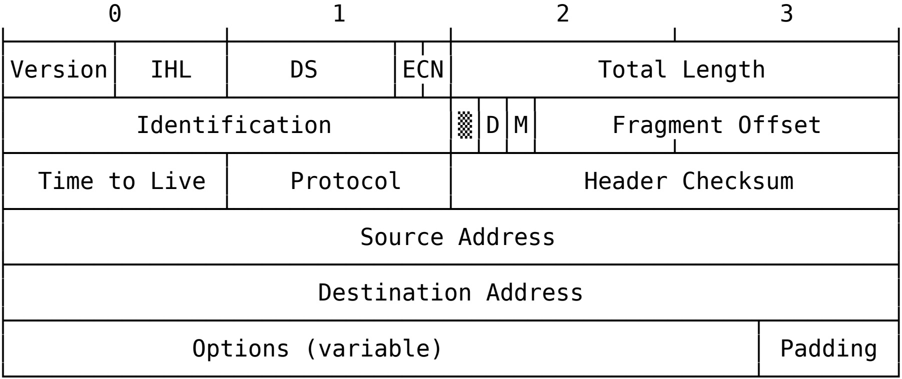


Alpha is sending an IP packet to epsilon…Please describe.

# Managing The Routes

- Manually maintained by administrator
- ICMP can report some routing problems
- For larger networks, routing protocols are used.

# IP Packet



```
        0               1               2               3
|---------------|---------------|---------------|---------------|
|Version|  IHL  |     DS    |ECN|          Total Length         |
|-------------------------------|-------------------------------|
|         Identification        |▒|D|M|    Fragment Offset      |
|-------------------------------|-------------------------------|
|  Time to Live |   Protocol    |        Header Checksum        |
|---------------------------------------------------------------|
|                         Source Address                        |
|---------------------------------------------------------------|
|                      Destination Address                      |
|---------------------------------------------------------------|
|                Options (variable)                  | Padding  |
|---------------------------------------------------------------|
```

# IP References

Wikipedia. *Internet Protocol — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Internet%5C_Protocol&oldid=645719875.

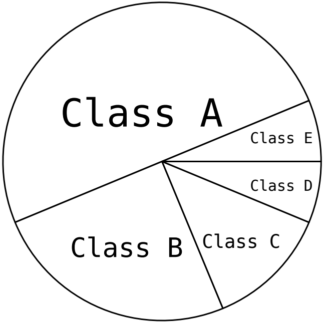
Wikipedia. *IP address — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=IP%5C_address&oldid=651153507.

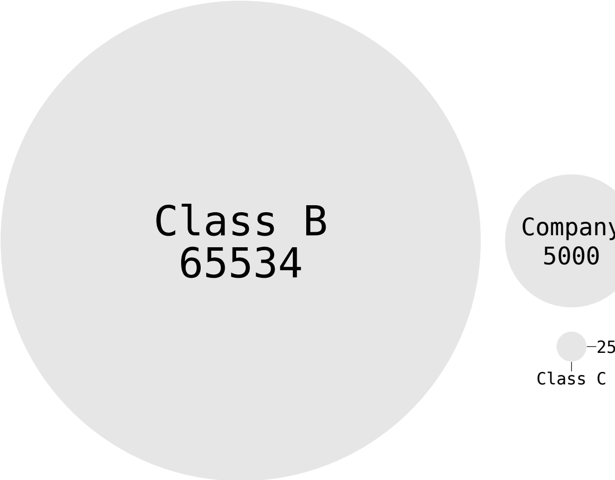
POSTEL J. *Internet Protocol*. RFC Editor. 1981. https://www.rfc-editor.org/rfc/rfc791.txt.

Wikipedia. *IPv4 header checksum — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=IPv4%5C_header%5C_checksum&oldid=645516564.

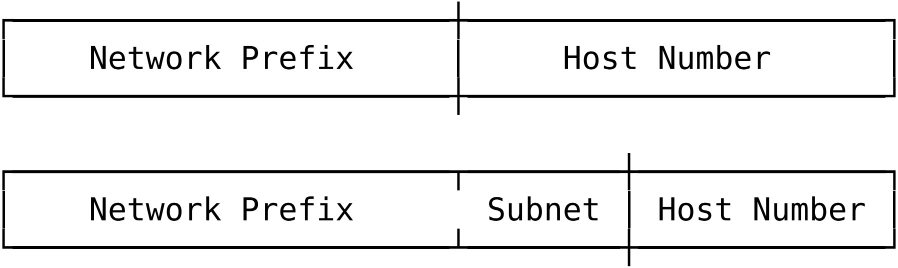
# Why Subnetting?

- save address space
- restrict collision domain
- security
- physical media (Ethernet, FDDI, …)





# How

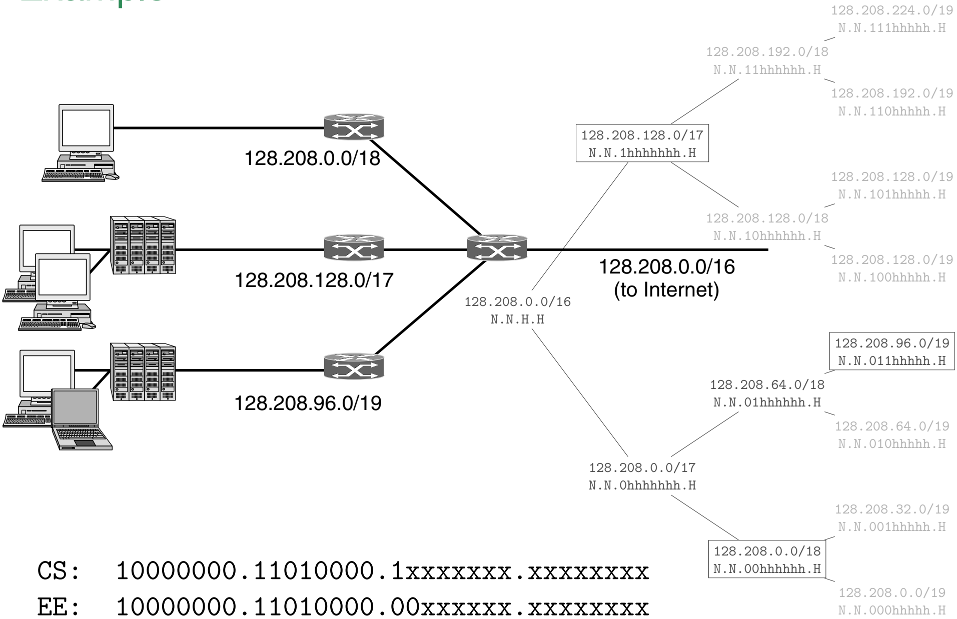


**Subnet mask** is a bitmask used to identify the network and node parts of the address.

## Default Subnet Masks

| | | |
|---|---|---|
| Class A | `255.0.0.0` | `11111111.0.0.0` |
| Class B | `255.255.0.0` | `11111111.11111111.0.0` |
| Class C | `255.255.255.0` | `11111111.11111111.11111111.0` |

# Example



```
 CS:   10000000.11010000.1xxxxxxx.xxxxxxxx
 EE:   10000000.11010000.00xxxxxx.xxxxxxxx
ART:   10000000.11010000.011xxxxx.xxxxxxxx
```

# $2^n - 2$

## Example

| | |
|---|---|
| IP address: | 11001010.11001011.10000100.11110001<br>202 . 203 . 132 . 241 |
| Subnet mask: | 11111111.11111111.11111111.11000000<br>255 . 255 . 255 . 192 |

- There are $2^2 - 2 = 2$ subnets
- Each subnet has $2^6 - 2 = 62$ nodes
- Subtract 2? All “0”s and all “1”s. (old story)

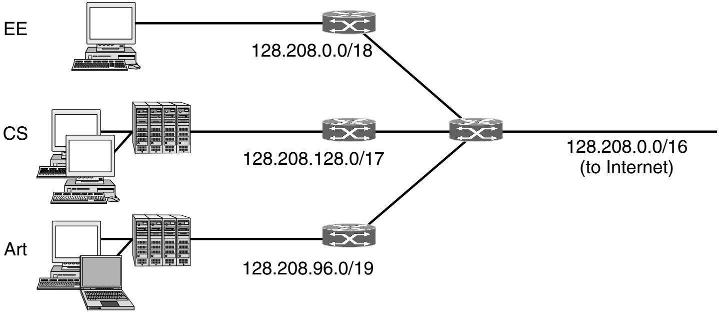
# Subnet Calculator

### free IP subnet calculators

Try:

```
~$ subnetcalc 202.203.132.244/26
~$ ipcalc 202.203.132.244/26
~$ sipcalc 202.203.132.244/26
```

# Quiz



Consider a packet addressing `128.208.2.251`

Q1: Which subnet it belongs to?

Q2: The route table inside each router?

# Subnetting References

- Wikipedia. *Subnetwork — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Subnetwork&oldid=645854808.

- Wikipedia. *IPv4 subnetting reference — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=IPv4%5C_subnetting%5C_reference&oldid=652583338.

- Wikipedia. *Private network — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Private%5C_network&oldid=649931709.

- MOGUL J. *Internet subnets*. RFC Editor. 1984. https://www.rfc-editor.org/rfc/rfc917.txt.

- MOGUL J, POSTEL J. *Internet Standard Subnetting Procedure*. RFC Editor. 1985. https://www.rfc-editor.org/rfc/rfc950.txt.

# CIDR—Classless Inter-Domain Routing

CIDR An IP addressing scheme that replaces the older system based on classes A, B and C.

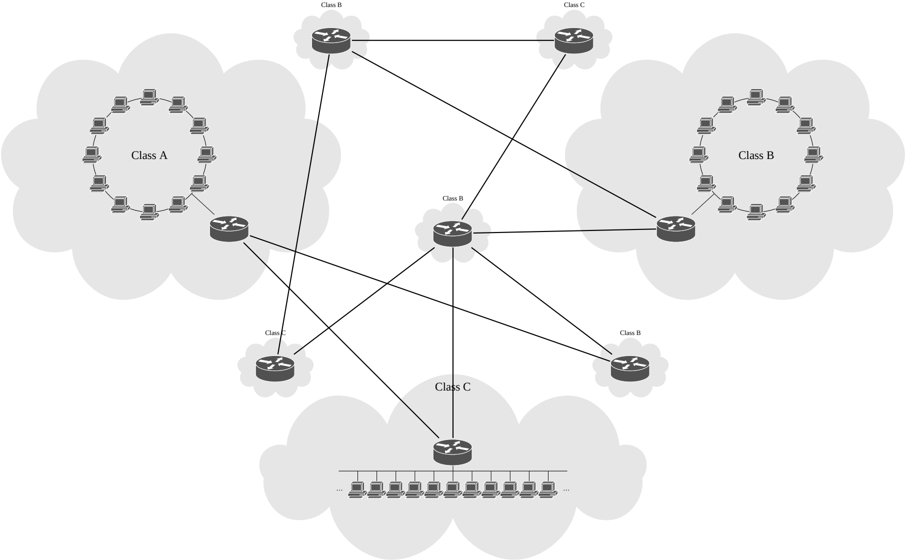
## Why?

With a new network being connected to the Internet every 30 minutes the Internet was faced with two critical problems:

- Running out of IP addresses
- Running out of capacity in the global routing tables

# Two-Level Internet

## Network – Host



# Running out of IP addresses

Using the old addressing scheme, the Internet could support:

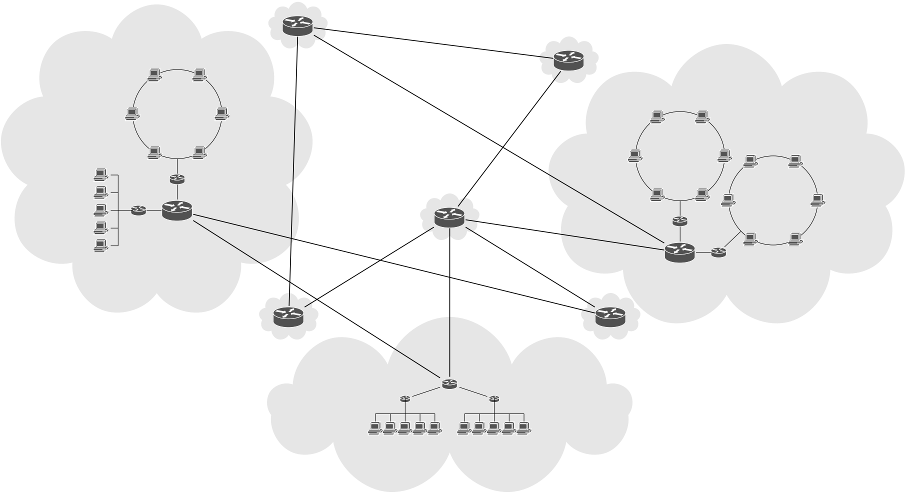
- 126 Class A networks that could include up to 16,777,214 hosts each
- Plus 65,000 Class B networks that could include up to 65,534 hosts each
- Plus over 2 million Class C networks that could include up to 254 hosts each

only 3% of the assigned addresses were actually being used.

# Global Routing Tables At Capacity

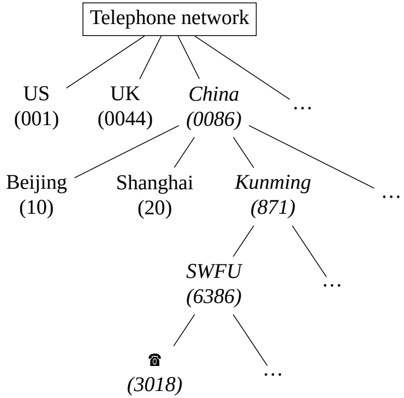
- As the number of networks on the Internet increased, so did the number of routes.
- A few years back it was forecasted that the global backbone Internet routers were fast approaching their limit on the number of routes they could support.
- Even using the latest router technology, the maximum theoretical routing table size is approximately 60,000 routing table entries.
- If nothing was done the global routing tables would have reached capacity by mid-1994 and all Internet growth would be halted.

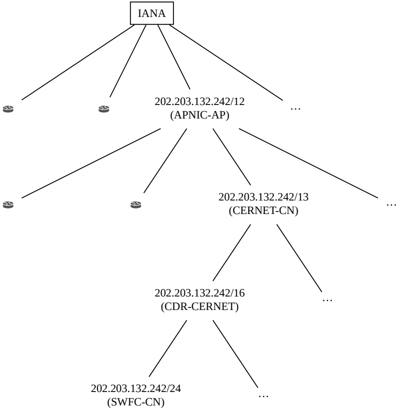
# Multi-Level Internet



# Hierarchical Routing Aggregation To Minimize Routing Table Entries

**Route Aggregation** a single high-level route entry can represent many lower-level routes in the global routing tables. Similar to the telephone network.



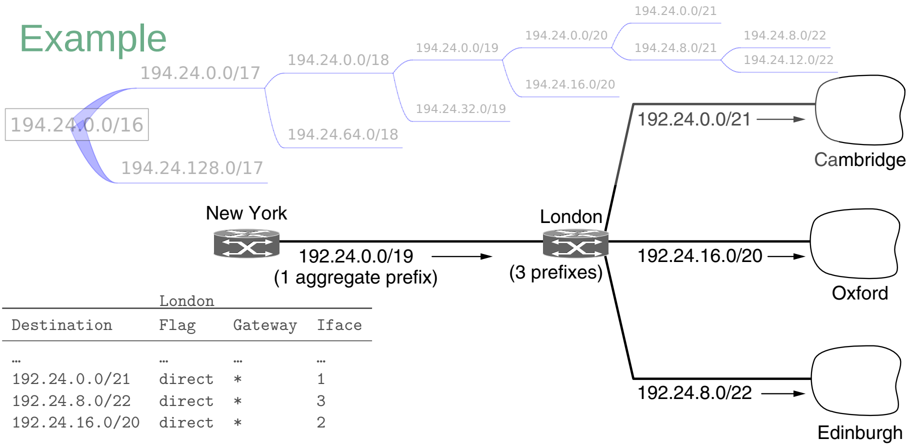


# Restructuring IP Address Assignments

Instead of being limited to network identifiers (or ”prefixes”) of 8, 16 or 24 bits, CIDR currently uses prefixes anywhere from 13 to 27 bits.

| | | |
|---|---|---|
| /27 | 1/8 of a Class C | 32 hosts |
| /26 | 1/4 of a Class C | 64 hosts |
| /25 | 1/2 of a Class C | 128 hosts |
| /24 | 1 Class C | 256 hosts |
| /16 | 256 Class C<br>(= 1 Class B) | 65,536 hosts |
| /13 | 2,408 Class C | 524,288 hosts |

# Example



London

| Destination | Flag | Gateway | Iface |
|---|---|---|---|
| … | … | … | … |
| 192.24.0.0/21 | direct | * | 1 |
| 192.24.8.0/22 | direct | * | 3 |
| 192.24.16.0/20 | direct | * | 2 |
| … | … | … | … |

# IP address assignments

| University | First address | Last address | How many | Prefix |
|---|---|---|---|---|
| Cambridge | 192.24.0.0 | 192.24.7.255 | 2048 | 192.24.0.0/21 |
| Edinburgh | 192.24.8.0 | 192.24.11.255 | 1024 | 192.24.8.0/22 |
| (Available) | 192.24.12.0 | 192.24.15.255 | 1024 | 192.24.12.0/22 |
| Oxford | 192.24.16.0 | 192.24.31.255 | 4096 | 192.24.16.0/20 |

## The Router in New York

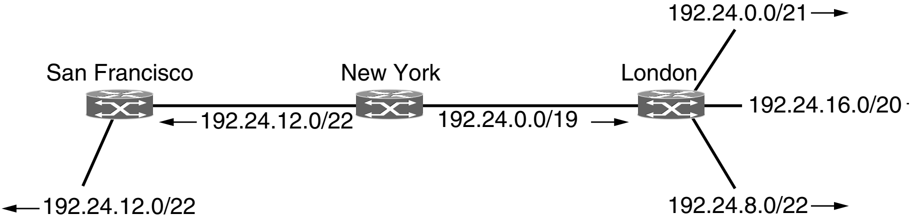
Option 1:

| Destination | Flag | Gateway | Interface |
|---|---|---|---|
| ... | ... | ... | ... |
| 192.24.0.0/21 | indirect | London | 1 |
| 192.24.8.0/22 | indirect | London | 1 |
| 192.24.16.0/20 | indirect | London | 1 |
| ... | ... | ... | ... |

Option 2:

| Destination | Flag | Gateway | Interface |
|---|---|---|---|
| ... | ... | ... | ... |
| 192.24.0.0/19 | direct | * | 1 |
| ... | ... | ... | ... |

# Longest Matching Prefix



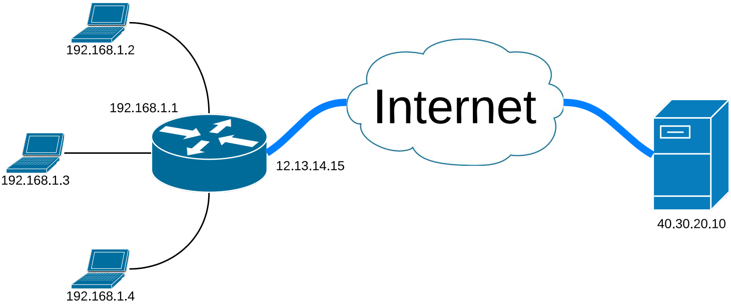
The router in New York

| Destination | Flag | Gateway | Interface |
|---|---|---|---|
| … | … | … | … |
| 192.24.0.0/19 | direct | * | 1 |
| 192.24.12.0/22 | direct | * | 2 |
| … | … | … | … |

# CIDR References

- Wikipedia. *Classless Inter-Domain Routing — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Classless%5C_Inter-Domain%5C_Routing&oldid=641285826.
- FULLER V, LI T. *Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan*. RFC Editor. 2006. https://www.rfc-editor.org/rfc/rfc4632.txt.

# Network Address Translation (NAT)



| **Src** | **NAT Router** |
|---|---|
| `IP:Port` | `IP:Port` |
| `192.168.1.2:3456` | `12.13.14.15:1` |
| `192.168.1.3:6789` | `12.13.14.15:2` |
| `192.168.1.3:8910` | `12.13.14.15:3` |
| `192.168.1.4:3750` | `12.13.14.15:4` |

# Why IPv6?

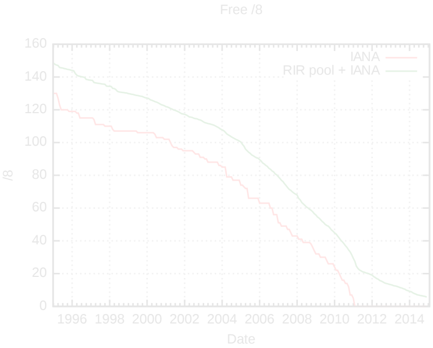
## No enough addresses!

Kidding? We have:

- $2^{32}$ address space
- NAT
- CIDR

No kidding. All gone.

- IANA: 31 January 2011
- Asia-Pacific: 15 April 2011
- Europe: 14 September 2012
- Latin America: 10 June 2014



So, we need a larger address space ($2^{128}$).

# $2^{128}$

## Why such a high number of bits?

For a larger address space

- Think about mobile phones, cars (inside devices), toasters, refrigerators, light switches, and so on…

## Why not higher?

- More bits ➠ bigger header ➠ more overhead
- max MTU on Ethernet is 1500 octets

| | min MTU (octets) | header length (octets) | overhead |
|---|---|---|---|
| IPv4 | 576 | 20~60 | 3.4% |
| IPv6 | 1280 | 40 | 3.8% |

## Why not IPv5?

4: is already used for IPv4

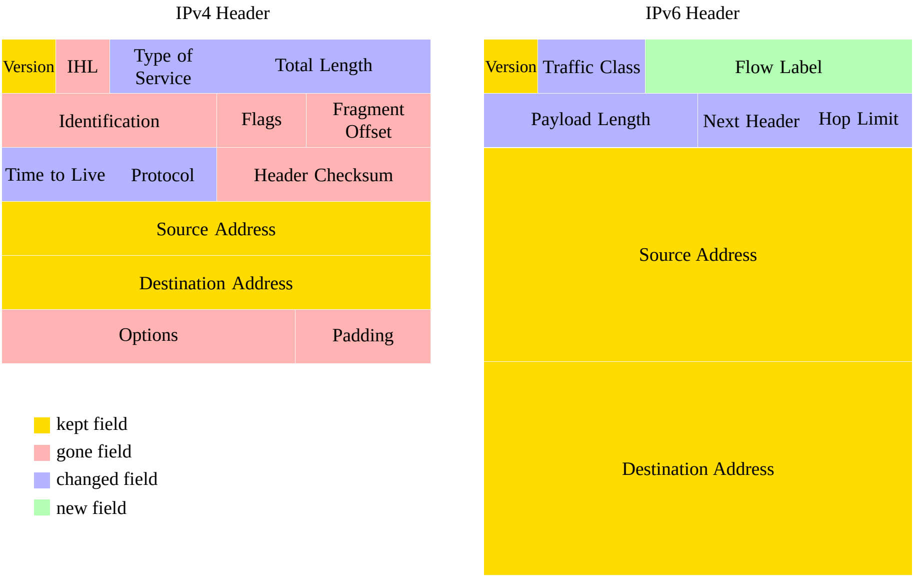
5: is reserved for the Stream Protocol (STP, RFC 1819 / Internet Stream Protocol Version 2) (which never really made it to the public)

6: The next free number. Hence IPv6 was born!

## More than a larger address space ($2^{128}$)

- Simplified header makes routing faster
- End-to-end connectivity
- Auto-configuration
- No broadcast
- Anycast
- Mobility — same IP address everywhere
- Network-layer security
- Extensibility
- and more ...

# Simplification



IPv4 Header

Version IHL Type of Service Total Length

Identification Flags Fragment Offset

Time to Live Protocol Header Checksum

Source Address

Destination Address

Options Padding

IPv6 Header

Version Traffic Class Flow Label

Payload Length Next Header Hop Limit

Source Address

Destination Address

- 🟨 kept field
- 🟥 gone field
- 🟦 changed field
- 🟩 new field

# IPv6 Extension Header

| | |
|---|---|
| IPv6 header<br>Next Header = TCP | TCP header + data |

| | | |
|---|---|---|
| IPv6 header<br>Next Header = Routing | Routing header<br>Next Header = TCP | TCP header + data |

| | | | |
|---|---|---|---|
| IPv6 header<br>Next Header = Routing | Routing header<br>Next Header = Fragment | Fragment header<br>Next Header = TCP | fragment of TCP header + data |

# IPv6 Addresses

## A real life address example

3ffe:ffff:0100:f101:0210:a4ff:fee3:9566

➥ 3ffe:ffff:100:f101:210:a4ff:fee3:9566

## More simplifications

3ffe:ffff:100:f101:0:0:0:1

➥ 3ffe:ffff:100:f101::1

## The biggest simplification

IPv6 localhost address

0000:0000:0000:0000:0000:0000:0000:0001 ➠ ::1

# Address types

Global unicast addresses begin with `[23]xxx`

e.g. `2001:db8:85a3::8a2e:370:7334`

Unique local addresses begin with `fc00::/7`

e.g. `fdf8:f53b:82e4::53`

Similiar to private IPs in IPv4

Link local addresses begin with `fe80::/64`

e.g. `fe80::62d8:19ff:fece:44f6/64`

Similiar to `169.254.0.0/16`

Localhost address `::1`

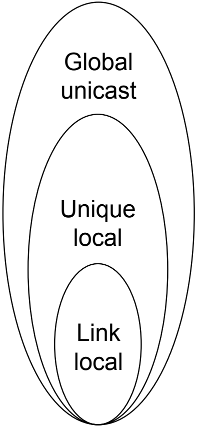
Similiar to IPv4 with its “`127.0.0.1`”

Multicast addresses begin with `ffxy::/8`

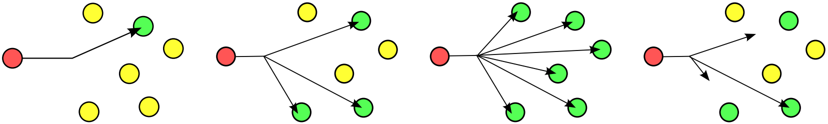
e.g. `ff01::2`

Unspecified address `::`

- Like “`any`” or “`0.0.0.0`” in IPv4



# Anycast addresses



Unicast
1-to-1

Multicast
1-to-n

Broadcast
1-to-all

Anycast
1-to-$\frac{1}{n}$

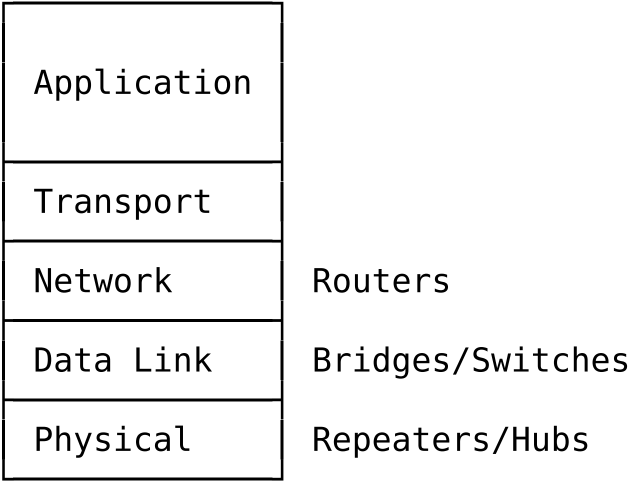
## Anycast

- is assigned to more than one interface
- a packet sent to an anycast address is routed to the ”nearest” interface having that address
- is allocated from the unicast address space

# IPv6 References

- Wikipedia. *IPv6 — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=IPv6&oldid=648071002.
- Wikipedia. *IPv6 packet — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=IPv6%5C_packet&oldid=651199118.
- Wikipedia. *IPv6 address — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=IPv6%5C_address&oldid=645435688.
- DEERING S, HINDEN R. *Internet Protocol, Version 6 (IPv6) Specification*. RFC Editor. 1998. https://www.rfc-editor.org/rfc/rfc2460.txt.
- HINDEN R, DEERING S. *IP Version 6 Addressing Architecture*. RFC Editor. 2006. https://www.rfc-editor.org/rfc/rfc4291.txt.

# Networking Devices



| | |
|---|---|
| Application | |
| Transport | |
| Network | Routers |
| Data Link | Bridges/Switches |
| Physical | Repeaters/Hubs |

# Repeater, Hub

Repeater connects network segments at the physical layer.

Hub a multi-port repeater

- simple, cheap
- Repeaters/Hubs do NOT isolate collision domains.
- 100m maximum

# Bridge, Switch

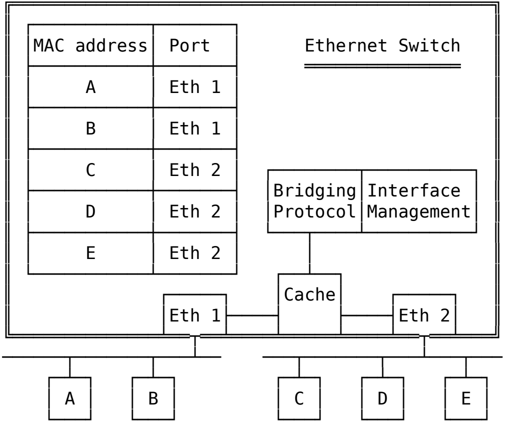
**Bridge** connects multiple network segments at the data link layer (layer 2)

**Switch** a multi-port bridge

## Transparent bridging

Uses a forwarding database to send frames across network segments

- Learning
- Flooding
- Forwarding
- Filtering
- Aging



Ethernet Switch

| MAC address | Port |
|---|---|
| A | Eth 1 |
| B | Eth 1 |
| C | Eth 2 |
| D | Eth 2 |
| E | Eth 2 |

Bridging Protocol

Interface Management

Cache

Eth 1

Eth 2

A

B

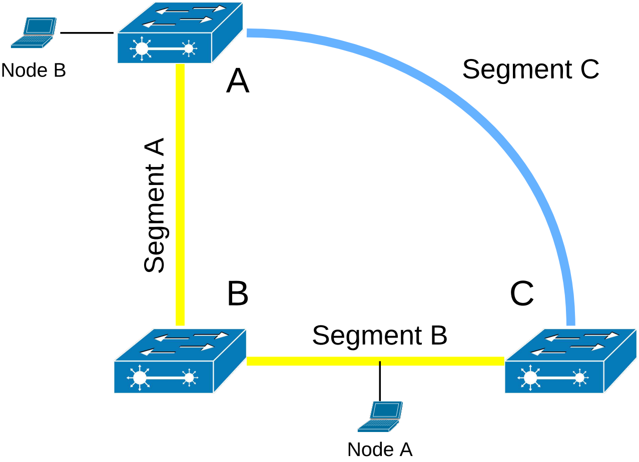
C

D

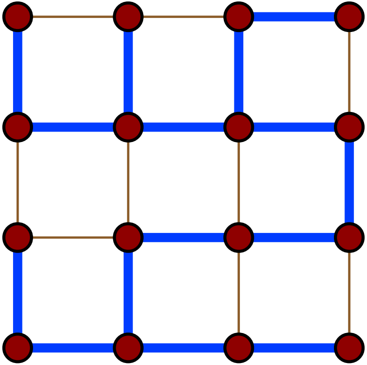
E

# Switch Loop

- ☺ Redundancy — Eliminating the single point of failure
- ☹ Broadcast storm — Resulting in potentially severe network congestion



Spanning Tree Protocol (STP) is a network protocol that ensures a loop-free topology for any bridged Ethernet local area network.



# Algorhyme

*I think that I shall never see*
*A graph more lovely than a tree.*

*A tree whose crucial property*
*Is loop-free connectivity.*

*A tree that must be sure to span*
*So packets can reach every LAN.*

*First, the root must be selected.*
*By ID, it is elected.*

*Least cost paths from root are traced.*
*In the tree, these paths are placed.*

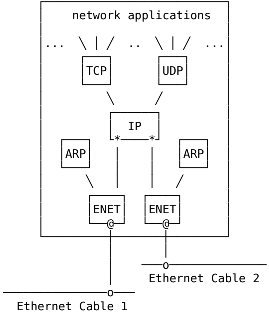
*A mesh is made by folks like me*
*Then bridges find a spanning tree.*

— Radia Perlman

# Router

Router connects two or more logical subnets at the network layer (layer 3)

Routing is to find a route in the route table



## Bridging vs. Routing

- A switch connects devices to create a network
- A router connects networks

| Bridging | Routing |
|---:|:---|
| L2 | L3 |
| MAC addr.(local) | IP addr.(global) |
| intranet | internet |
| Forwarding DB | Routing table |
| relearn, flooding | more efficient |

- to put multiple segments into one bridged network, or
- to divide it into different networks interconnected by routers

# More About Networking Devices

- Wikipedia. *Router (computing) — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Router%5C_(computing)&oldid=646784918.
- Wikipedia. *Routing table — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Routing%5C_table&oldid=644938703.
- Wikipedia. *Network switch — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Network%5C_switch&oldid=646928384.
- Wikipedia. *LAN switching — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=LAN%5C_switching&oldid=651228780.

# What’s A Packet Filter?

A packet filter is a piece of software which looks at the `header` of packets as they pass through, and decides the fate of the entire packet. It might decide to

- `DROP` the packet (i.e., discard the packet as if it had never received it),
- `ACCEPT` the packet (i.e., let the packet go through), or
- something more complicated.

# Packet Filter Under Linux

iptables talks to the kernel and tells it what packets to filter.

The iptables tool inserts/deletes rules from the kernel's packet filtering table.

# Quick Start

Debian/Ubuntu users can do:

```
~$ sudo apt install iptables
~$
~$ sudo iptables -A INPUT -s 147.8.212.123 -p all -j DROP
~$
~$ sudo iptables -D INPUT -s 147.8.212.123 -p all -j DROP
~$
~$ man iptables
~$
~$ chromium http://www.netfilter.org/documentation/
~$
```

# Terminology

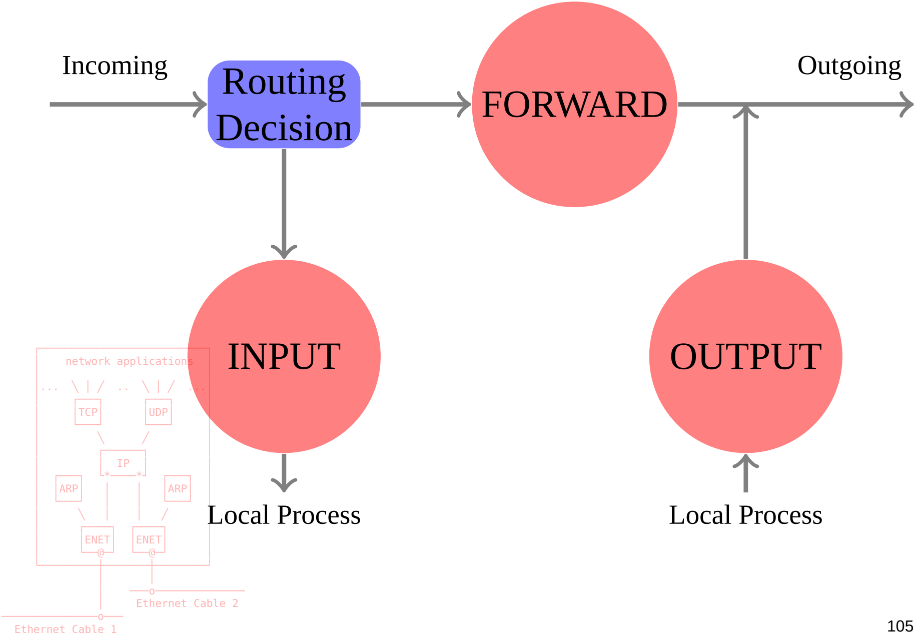
Filter table is in the kernel, contains `chains`.

Chains a.k.a. firewall chains, are lists of filtering rules. The three kernel built-in chains are called `INPUT`, `OUTPUT`, and `FORWARD`.

Rules Each rule says:

`if` the packet header looks like this
`then` here's what to do with the packet

# How Chains Work?



# Using iptables

To manage whole chains:

1. Create a new chain (`-N`).
2. Delete an empty chain (`-X`).
3. Change the policy for a built-in chain. (`-P`).
4. List the rules in a chain (`-L`).
5. Flush the rules out of a chain (`-F`).
6. Zero the packet and byte counters on all rules in a chain (`-Z`).

To manipulate rules inside a chain:

1. Append a new rule to a chain (`-A`).
2. Insert a new rule at some position in a chain (`-I`).
3. Replace a rule at some position in a chain (`-R`).
4. Delete a rule at some position in a chain, or the first that matches (`-D`).

# Examples

```
~$ ping -c 1 127.0.0.1
~$
~$ sudo iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP
~$
~$ ping -c 1 127.0.0.1
~$
~$ sudo iptables -D INPUT -s 127.0.0.1 -p icmp -j DROP
~$
~$ sudo iptables -A INPUT -s ! 127.0.0.1 -p all -j DROP
~$
~$ sudo iptables -A INPUT -s 192.168.1.0/24 -p all -j DROP
~$
```

# More Examples

```
~$ # Syn-flood protection:
~$ sudo iptables -A FORWARD -p tcp --syn -m limit --limit 1/s -j ACCEPT
~$
~$ # Furtive port scanner:
~$ sudo iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s -j ACCEPT
~$
~$ # Ping of death:
~$ sudo iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --limit 1/s -j ACCEPT
~$
```

# NAT & Packet Filtering References

- Wikipedia. *Network address translation — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Network%5C_address%5C_translation&oldid=652836698.

- EGEVANG K, FRANCIS P. *The IP Network Address Translator (NAT)*. RFC Editor. 1994. https://www.rfc-editor.org/rfc/rfc1631.txt.

- TYSON J. *How Network Address Translation Works — HowStuffWorks.com*. 2001. %5Curl%7Bhttp://computer.howstuffworks.com/nat.htm%7D.

- CONTRIBUTORS W. *Iptables — Wikipedia, The Free Encyclopedia*. 2017. https://en.wikipedia.org/w/index.php?title=Iptables&oldid=817424711.

# ☎ vs. ✉

## Circuit switching (☎)

- ☺ guaranteed performance
- ☺ fast transfers (once circuit is established)
- ☹ wastes bandwidth if traffic is "bursty"
- ☹ connection setup adds delay
- ☹ recovery from failure is slow

## Packet switching (✉)

- ☹ no guaranteed performance
- ☹ header overhead per packet
- ☹ queues and queuing delay
- ☺ efficient use of bandwidth
- ☺ no connection setup
- ☺ can "route around trouble"

IP: host ↔ host

TCP/UDP: process ↔ process

## IP provides *unreliable* service

*Best-effort, no guarantee*

- ? segment delivery
- ? orderly delivery of segments
- ? the integrity of the data in the segments

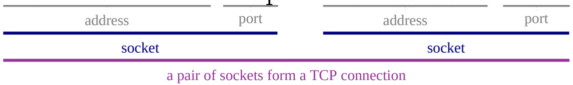
## TCP provides *reliable* data transfer

*You receive none or you receive it correctly and orderly*.

- ✓ correctness — acknowledgement, checksum
- ✓ order — sequence numbers
- ✓ packet lost — timers
- ✓ flow control — sliding window
- ✓ congestion control

# A TCP Connection

```
wx672@cs3:~$ netstat -at | grep http | grep ESTAB
tcp   0   0 cs3.swfu.edu.cn:http   220.163.96.3:47179   ESTABLISHED
```



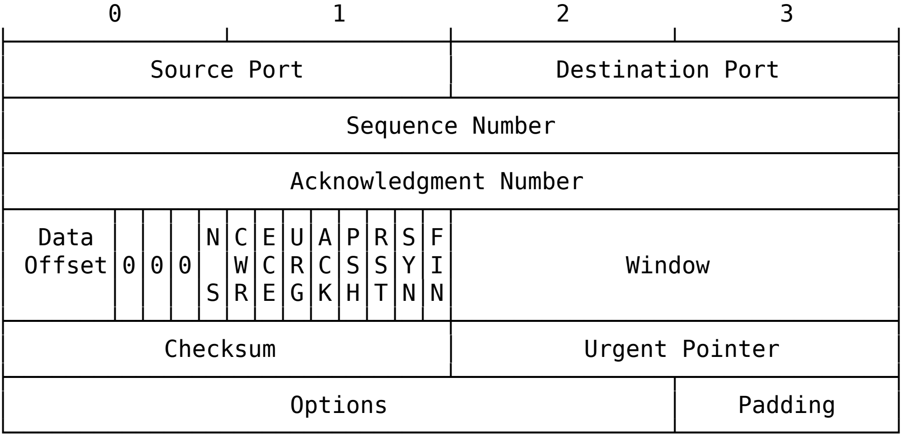
## Port numbers

port range: 0 ~ 65535

well-known ports: 0 ~ 1023

| | | | | | |
|---|---|---|---|---|---|
| FTP | 20/21 | SSH | 22 | Telnet | 23 |
| SMTP | 25 | DNS | 53 | DHCP | 67/68 |
| HTTP | 80 | POP3 | 110 | HTTPS | 443 |
| IMAP4 | 143 | | | | |

# TCP Header



0 1 2 3

Source Port | Destination Port

Sequence Number

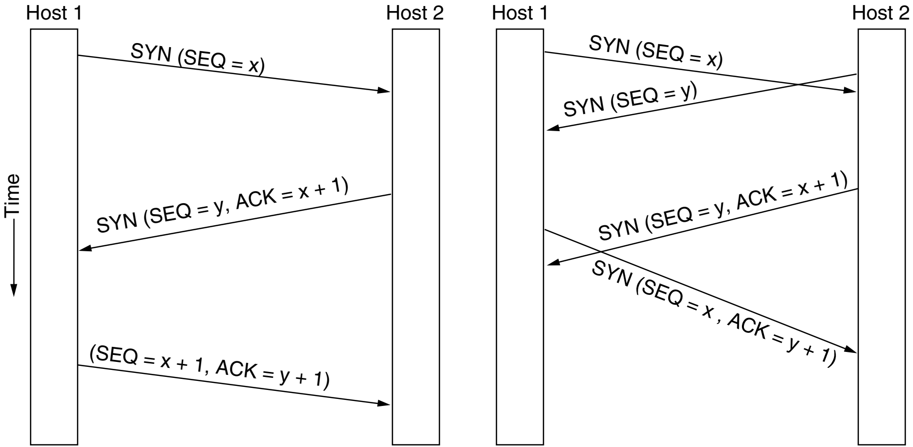
Acknowledgment Number

Data Offset | 0 | 0 | 0 | NS | CWR | ECE | URG | ACK | PSH | RST | SYN | FIN | Window

Checksum | Urgent Pointer

Options | Padding

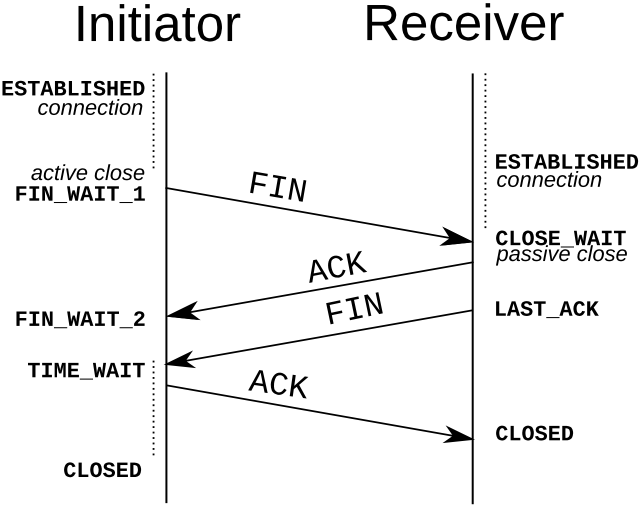
# Establishing a TCP Connection

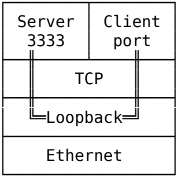

Host 1
Host 2
Time
SYN (SEQ = x)
SYN (SEQ = y, ACK = x + 1)
(SEQ = x + 1, ACK = y + 1)
Host 1
Host 2
SYN (SEQ = x)
SYN (SEQ = y)
SYN (SEQ = y, ACK = x + 1)
SYN (SEQ = x , ACK = y + 1)

(a)

(b)

# Closing a TCP Connection



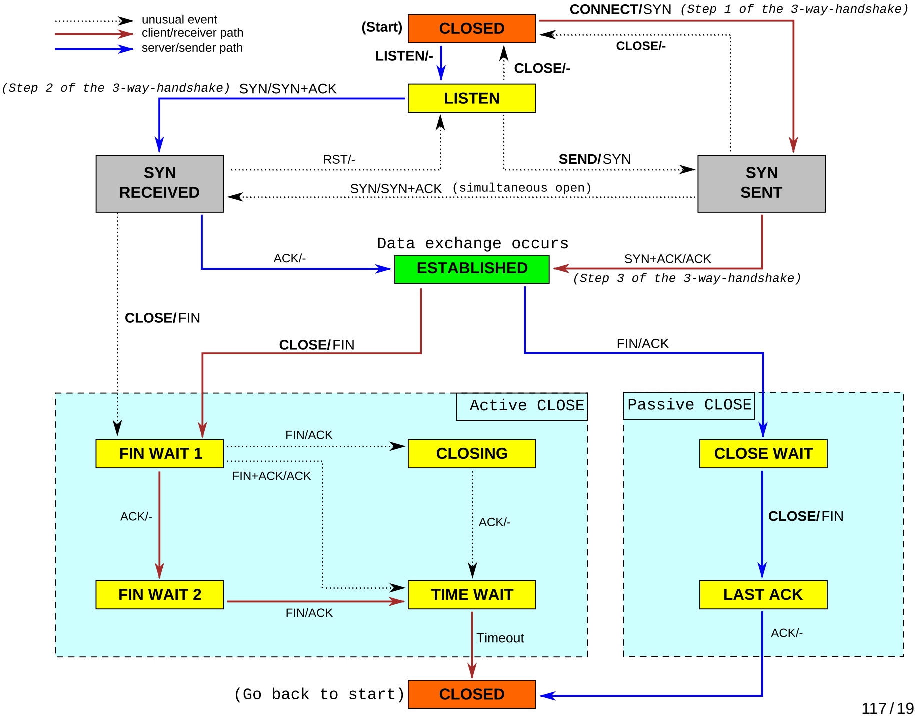


- Terminal A: `nc -l 3333`
- Terminal B: `nc localhost 3333`

## tcpdump output in terminal C:

```
~$ sudo tcpdump -i lo -S port 3333
```

```
12:47:09.106903 IP localhost.37831 > localhost.3333:
  Flags [S], seq 2485057335, win 32792, ..., length 0
12:47:09.106923 IP localhost.3333 > localhost.37831:
  Flags [S.], seq 2476477986, ack 2485057336, win 32768, ..., length 0
12:47:09.106936 IP localhost.37831 > localhost.3333:
  Flags [.], ack 2476477987, win 257, ..., length 0
12:47:26.963149 IP localhost.37831 > localhost.3333:
  Flags [F.], seq 2485057336, ack 2476477987, win 257, ..., length 0
12:47:26.963244 IP localhost.3333 > localhost.37831:
  Flags [F.], seq 2476477987, ack 2485057337, win 256, ..., length 0
12:47:26.963264 IP localhost.37831 > localhost.3333:
  Flags [.], ack 2476477988, win 257, ..., length 0
```



unusual event
client/receiver path
server/sender path

(Start) CLOSED

CONNECT/SYN *(Step 1 of the 3-way-handshake)*

CLOSE/-

LISTEN/-

CLOSE/-

LISTEN

*(Step 2 of the 3-way-handshake)* SYN/SYN+ACK

RST/-

SEND/SYN

SYN RECEIVED

SYN SENT

SYN/SYN+ACK (simultaneous open)

Data exchange occurs

ACK/-

ESTABLISHED

SYN+ACK/ACK

*(Step 3 of the 3-way-handshake)*

CLOSE/FIN

CLOSE/FIN

FIN/ACK

Active CLOSE

Passive CLOSE

FIN WAIT 1

FIN/ACK

CLOSING

CLOSE WAIT

FIN+ACK/ACK

ACK/-

ACK/-

CLOSE/FIN

FIN WAIT 2

TIME WAIT

LAST ACK

FIN/ACK

Timeout

ACK/-

(Go back to start) CLOSED

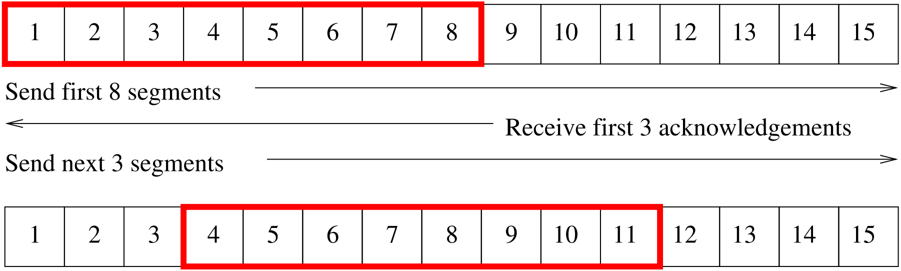
# netstat

```
~$ netstat -ant          ~$ netstat -ie
~$ netstat -antp         ~$ netstat -antp | grep ESTAB
~$ netstat -antpe        ~$ netstat -nlp | grep :80
~$ netstat -nr           ~$ man netstat
```

ss a netstat replacement. Example:

```
~$ ss -4ant "( sport = 3333 or dport = 3333 )"
~$ man ss
```

# Sliding Window



1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Send first 8 segments →

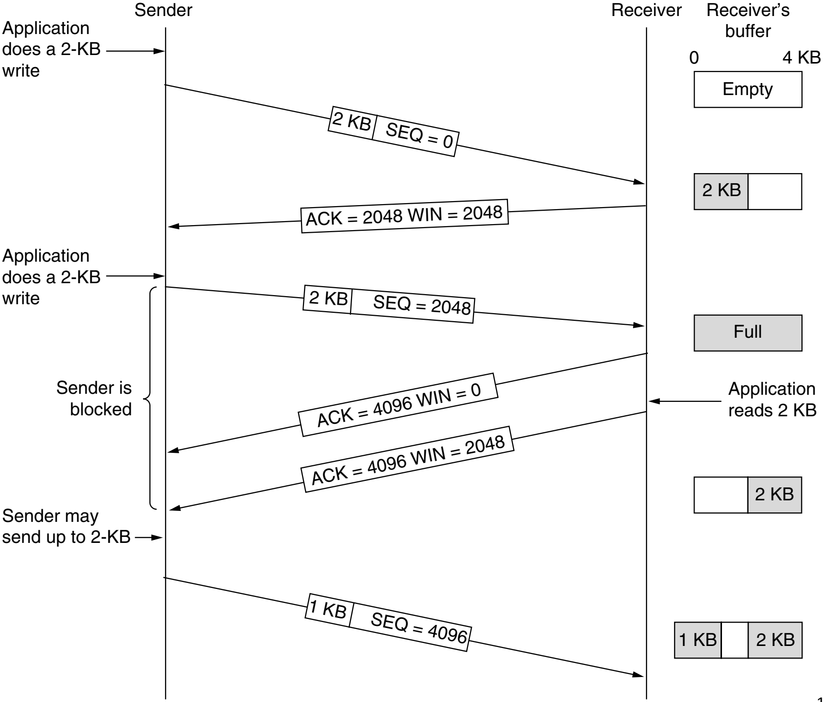
← Receive first 3 acknowledgements

Send next 3 segments →

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

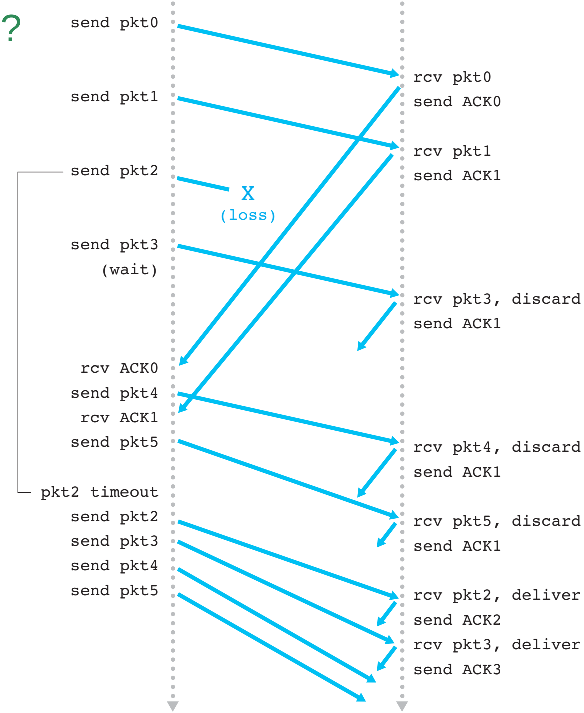
## The sliding window serves several purposes:

- guarantees the reliable delivery of data
- ensures that the data is delivered in order
- enforces flow control between the sender and the receiver.


Sender
Receiver
Receiver's buffer
0
4 KB
Application does a 2-KB write
Empty
2 KB SEQ = 0
2 KB
ACK = 2048 WIN = 2048
Application does a 2-KB write
2 KB SEQ = 2048
Full
Sender is blocked
ACK = 4096 WIN = 0
Application reads 2 KB
ACK = 4096 WIN = 2048
2 KB
Sender may send up to 2-KB
1 KB SEQ = 4096
1 KB
2 KB

# Packet Lost?

## Go-Back-N



send pkt0

rcv pkt0

send ACK0

send pkt1

rcv pkt1

send ACK1

send pkt2

X

(loss)

send pkt3

(wait)

rcv pkt3, discard

send ACK1

rcv ACK0

send pkt4

rcv ACK1

send pkt5

rcv pkt4, discard

send ACK1

pkt2 timeout

rcv pkt5, discard

send ACK1

send pkt2

send pkt3

send pkt4

send pkt5

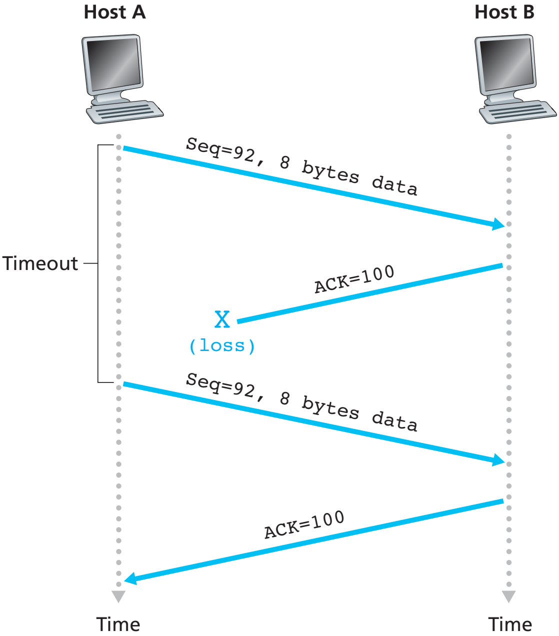
rcv pkt2, deliver

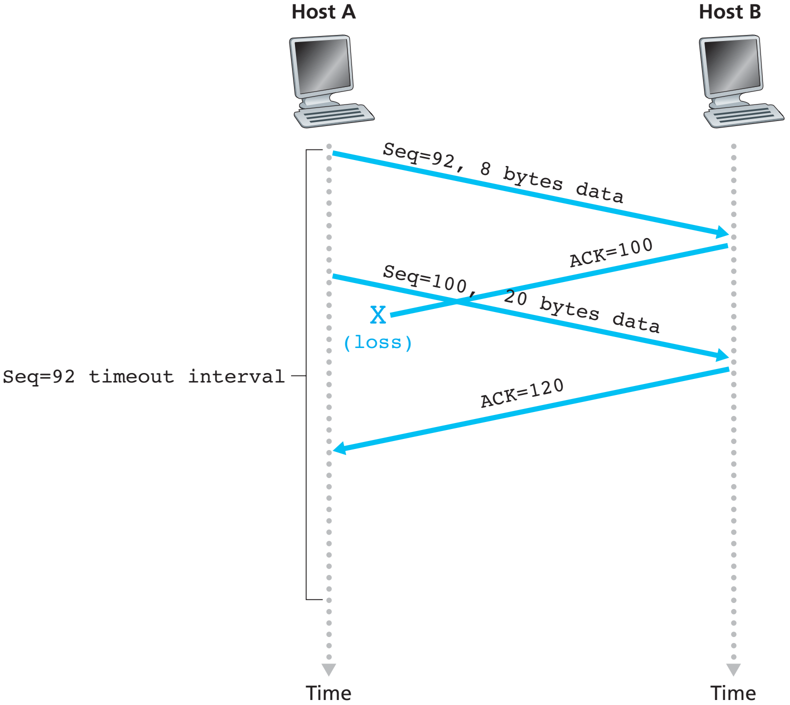
send ACK2

rcv pkt3, deliver

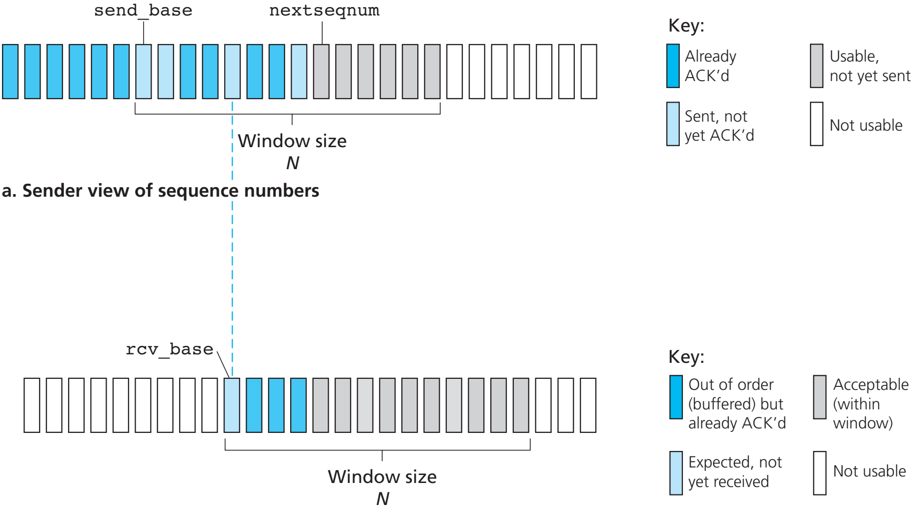
send ACK3

# ACK lost?


Host A
Host B
Seq=92, 8 bytes data
Timeout
ACK=100
X
(loss)
Seq=92, 8 bytes data
ACK=100
Time
Time

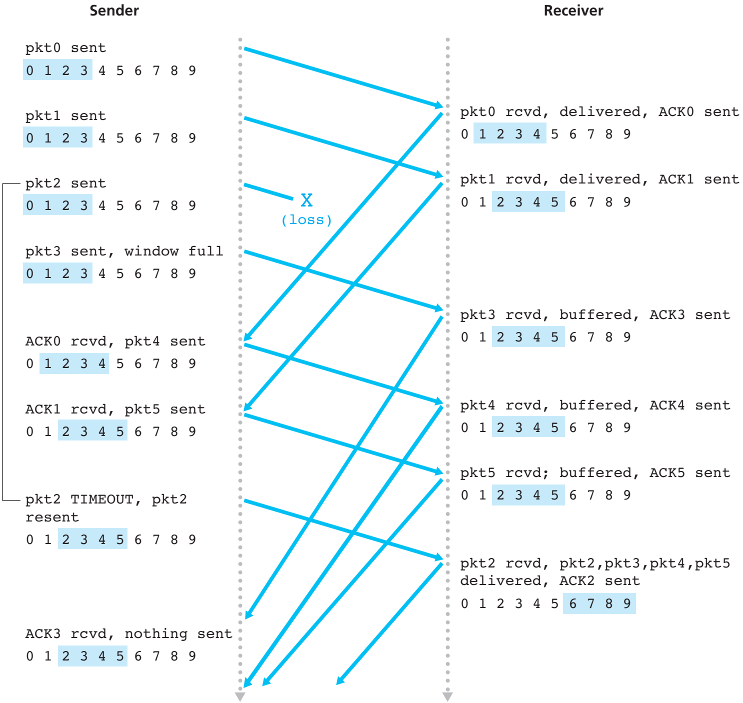

Host A
Host B
Seq=92, 8 bytes data
ACK=100
Seq=100, 20 bytes data
X
(loss)
Seq=92 timeout interval
ACK=120
Time
Time

# Selective-repeat

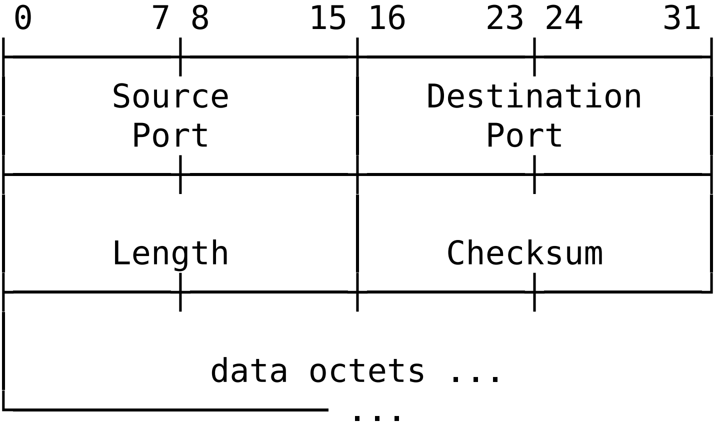


**a. Sender view of sequence numbers**

**b. Receiver view of sequence numbers**


Sender
Receiver
pkt0 sent
0 1 2 3 4 5 6 7 8 9
pkt0 rcvd, delivered, ACK0 sent
0 1 2 3 4 5 6 7 8 9
pkt1 sent
0 1 2 3 4 5 6 7 8 9
pkt1 rcvd, delivered, ACK1 sent
0 1 2 3 4 5 6 7 8 9
pkt2 sent
0 1 2 3 4 5 6 7 8 9
X
(loss)
pkt3 sent, window full
0 1 2 3 4 5 6 7 8 9
pkt3 rcvd, buffered, ACK3 sent
0 1 2 3 4 5 6 7 8 9
ACK0 rcvd, pkt4 sent
0 1 2 3 4 5 6 7 8 9
pkt4 rcvd, buffered, ACK4 sent
0 1 2 3 4 5 6 7 8 9
ACK1 rcvd, pkt5 sent
0 1 2 3 4 5 6 7 8 9
pkt5 rcvd; buffered, ACK5 sent
0 1 2 3 4 5 6 7 8 9
pkt2 TIMEOUT, pkt2 resent
0 1 2 3 4 5 6 7 8 9
pkt2 rcvd, pkt2,pkt3,pkt4,pkt5 delivered, ACK2 sent
0 1 2 3 4 5 6 7 8 9
ACK3 rcvd, nothing sent
0 1 2 3 4 5 6 7 8 9

# UDP Datagram



0 7 8 15 16 23 24 31

Source Port

Destination Port

Length

Checksum

data octets ...

...

# TCP/UDP References

Wikipedia. *Transmission Control Protocol — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Transmission%5C_Control%5C_Protocol&oldid=647944260.

Wikipedia. *User Datagram Protocol — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=User%5C_Datagram%5C_Protocol&oldid=643803508.

Wikipedia. *Checksum — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Checksum&oldid=645584712.

POSTEL J. *Transmission Control Protocol*. RFC Editor. 1981. https://www.rfc-editor.org/rfc/rfc793.txt.

POSTEL J. *User Datagram Protocol*. RFC Editor. 1980. https://www.rfc-editor.org/rfc/rfc768.txt.

# UDPClient.py I

```
#!/usr/bin/python

from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = raw_input('Input lowercase sentence:')
clientSocket.sendto(message,(serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print modifiedMessage
clientSocket.close()
```

# UDPClient.py II

`socket(AF_INET, SOCK_DGRAM)`

- `AF_INET`: using IPv4
- `SOCK_DGRAM`: UDP socket
- `clientPort` will be generated automatically

`clientSocket.sendto(message,(serverName, serverPort))`

1. attaches both the destination address `(serverName, serverPort)` and the source address `(clientIP, clientPort)` to the message
2. send the message

# UDPClient.py III

```
modifiedMessage, serverAddress =
clientSocket.recvfrom(2048)
```

1. puts the received message data into `modifiedMessage`
2. puts the source address `(IP, Port)` into `serverAddress`

- `2048`: buffer size

# UDPServer.py

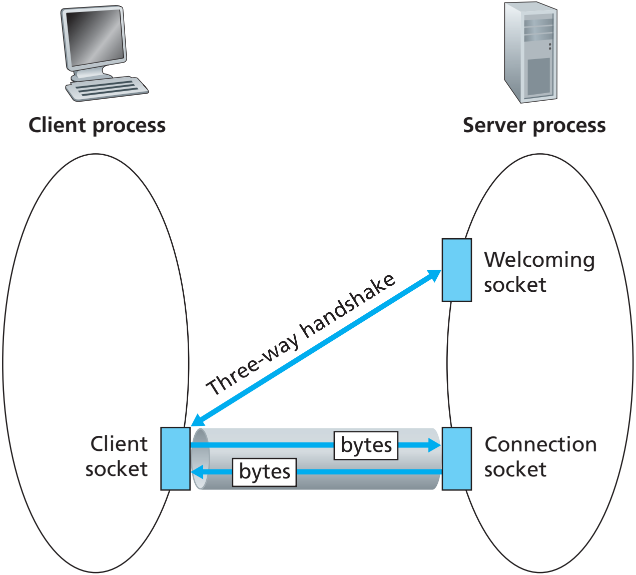
```python
#!/usr/bin/python

from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print "The server is ready to receive"
while 1:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage, clientAddress)
```

`serverSocket.bind(('', serverPort))`

- explicitly assigns 12000 to the server’s socket

# TCP Sockets

## Two Sockets at the Server



# TCPClient.py

```python
#!/usr/bin/python

#import time
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
print clientSocket.getsockname()
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence)
modifiedSentence = clientSocket.recv(1024)
print 'From Server:', modifiedSentence
#while 1:
#    time.sleep(1000)
clientSocket.close()
```

- `SOCK_STREAM`: TCP socket
- `connect()`: initiate the TCP connection (3-way handshake)
- `send()`: send out `sentence` through the client's socket. No destination address needs to be specified

# TCPServer.py

```python
#!/usr/bin/python

from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(5)
print serverSocket.getsockname()
print 'The server is ready to receive'
while 1:
    connectionSocket, addr = serverSocket.accept()
    print connectionSocket.getsockname()
    sentence = connectionSocket.recv(1024)
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
```

- `serverSocket`: the welcoming socket
- `connectionSocket`: a socket dedicated to this particular client
- `listen(backlog)`: the server listens for connection requests.
  - `backlog`: how many non-`accept()`-ed connections are allowed to be queueing
- `accept()`: whenever a connection request coming, creates a new `connectionSocket` (handshaking is done here)

# Homework

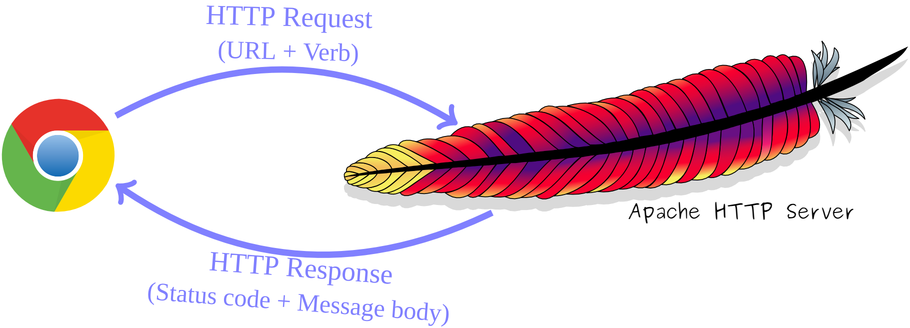
Re-write the above UDP/TCP client-server program in C.

# Socket References

Wikipedia. *Network socket — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Network%5C_socket&oldid=643452418.

HALL B. *Beej's Guide to Network Programming: Using Internet Sockets*. 2012.

# HTTP



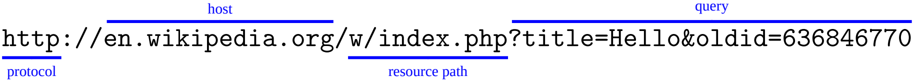
HTTP Request
(URL + Verb)

HTTP Response
(Status code + Message body)

Apache HTTP Server

# HTTP Request

## URL



host

query

```
http://en.wikipedia.org/w/index.php?title=Hello&oldid=636846770
```

protocol

resource path

```
~$ curl -v cs2.swfu.edu.cn/index.html
* Connected to cs2.swfu.edu.cn (202.203.132.242) port 80
> GET /index.html HTTP/1.1
> User-Agent: curl/7.38.0
> Host: cs2.swfu.edu.cn
> Accept: */*
>
```

Request line

Header lines

Empty line

## Verbs

GET POST PUT PATCH

HEAD OPTIONS DELETE TRACE CONNECT

# HTTP Response

```
< HTTP/1.1 200 OK
< Date: Thu, 15 Jan 2015 08:18:50 GMT
< Server: Apache/2.4.10 (Debian)
< Last-Modified: Tue, 02 Sep 2014 03:49:24 GMT
< ETag: "1fd-5020d015e5e4a"
< Accept-Ranges: bytes
< Content-Length: 509
< Vary: Accept-Encoding
< Content-Type: text/html
<
<html>
<head>
<title>Hello, world!</title>
</head>
<body>
<h1>Hello, world!</h1>
</body>
</html>
```

Status line

Header lines

Empty line

Data

# Status Codes

1xx Informational Messages

e.g. `104 Connection Reset by Peer`

2xx Successful

e.g. `200 OK`

3xx Redirection

e.g. `301 Moved Permanently`

4xx Client Error

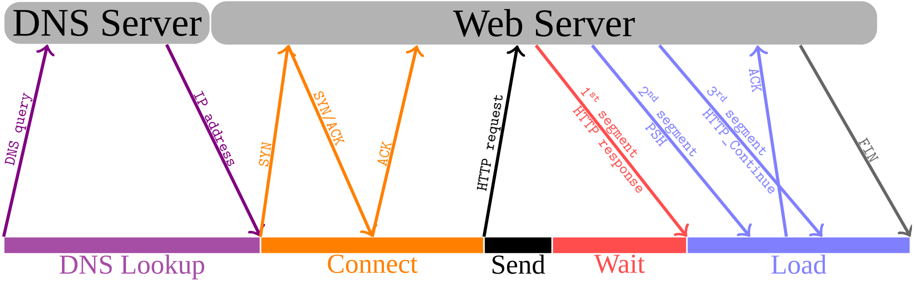
e.g. `404 Not Found`

5xx Server Error

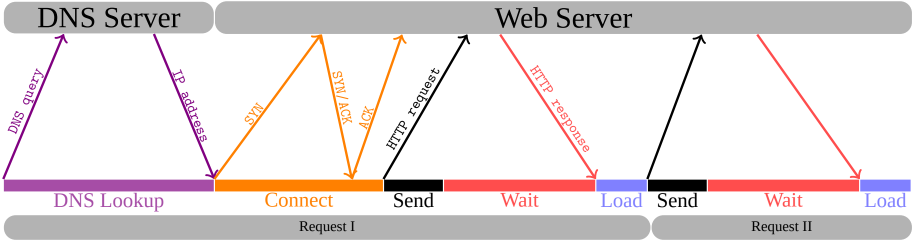
e.g. `500 Internal Server Error`

# HTTP Transaction

## Non-persistent — separate TCP connection



# Persistent — same TCP connection



# Stateless Protocol

A HTTP server maintains no information about the clients.

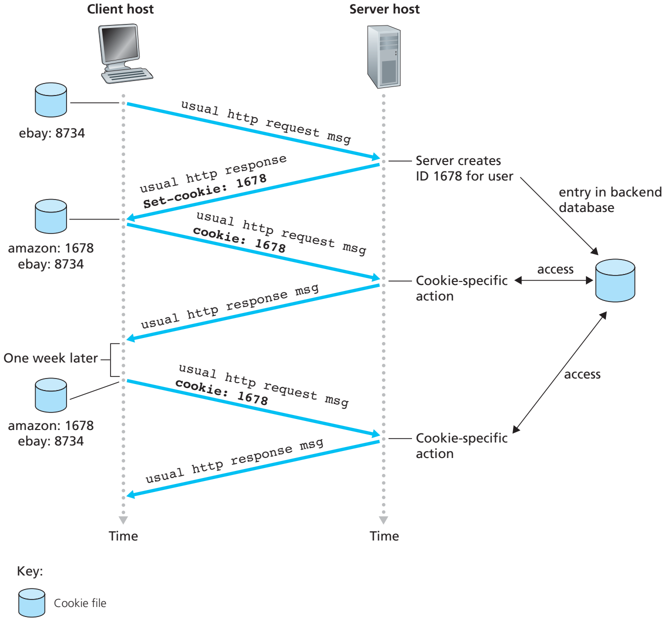
## Advantages

- Simplifies server design
- Save server resources (RAM...)
- Serve more users

## Disadvantages

- Missing information

# Keeping User State With Cookies



# HTTP/2

Quoted from http://http2.github.io/faq/

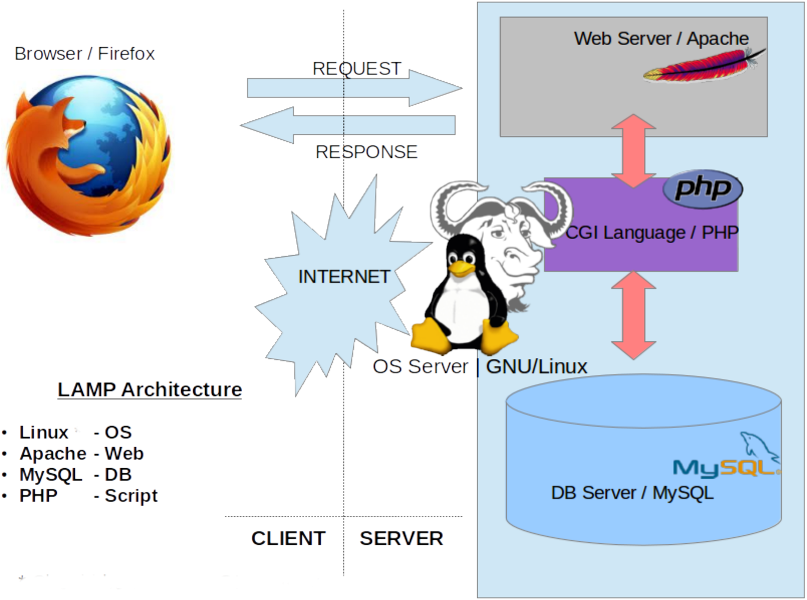
- is binary, instead of textual
- is fully multiplexed, instead of ordered and blocking
- can therefore use one connection for parallelism
- uses header compression to reduce overhead
- allows servers to “push” responses proactively into client caches

May 2015 Publish HTTP/2 as RFC7540/7541

# HTML

```html
<html>
    <head>
        <title>Hello, world!</title>
    </head>
    <body>
        <H1>Hello, world!</H1>
    </body>
</html>
```

LAMP
Linux Apache MySQL PHP,Perl,Python



Browser / Firefox

REQUEST

RESPONSE

INTERNET

Web Server / Apache

CGI Language / PHP

OS Server | GNU/Linux

DB Server / MySQL

CLIENT SERVER

## LAMP Architecture

- Linux - OS
- Apache - Web
- MySQL - DB
- PHP - Script

# HTTP References I

Wikipedia. *Hypertext Transfer Protocol — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Hypertext%5C_Transfer%5C_Protocol&oldid=648108367.

Wikipedia. *HTTP/2 — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=HTTP/2&oldid=648155546.

Wikipedia. *HTTP cookie — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=HTTP%5C_cookie&oldid=648216857.

Wikipedia. *Stateless protocol — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Stateless%5C_protocol&oldid=645610703.

Wikipedia. *HTML — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=HTML&oldid=648021866.

Wikipedia. *LAMP (software bundle) — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=LAMP%5C_(software%5C_bundle)&oldid=646364288.

# HTTP References II

FIELDING R, GETTYS J, MOGUL J, et al. *Hypertext Transfer Protocol – HTTP/1.1*. RFC Editor. 1999. https://www.rfc-editor.org/rfc/rfc2616.txt.

BELSHE M, PEON R, THOMSON (ED.) M. *Hypertext Transfer Protocol Version 2 (HTTP/2)*. RFC Editor. 2015. https://www.rfc-editor.org/rfc/rfc7540.txt.

PEON R, RUELLAN H. *HPACK: Header Compression for HTTP/2*. RFC Editor. 2015. https://www.rfc-editor.org/rfc/rfc7541.txt.

# DNS

## Names and Addresses

RFC 791, page 7:

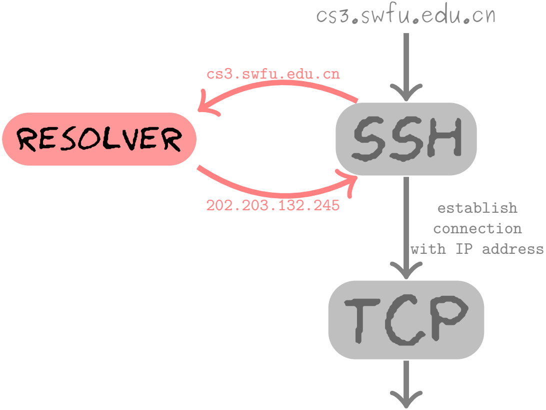
A name indicates what we seek

An address indicates where it is

A route indicates how to get there

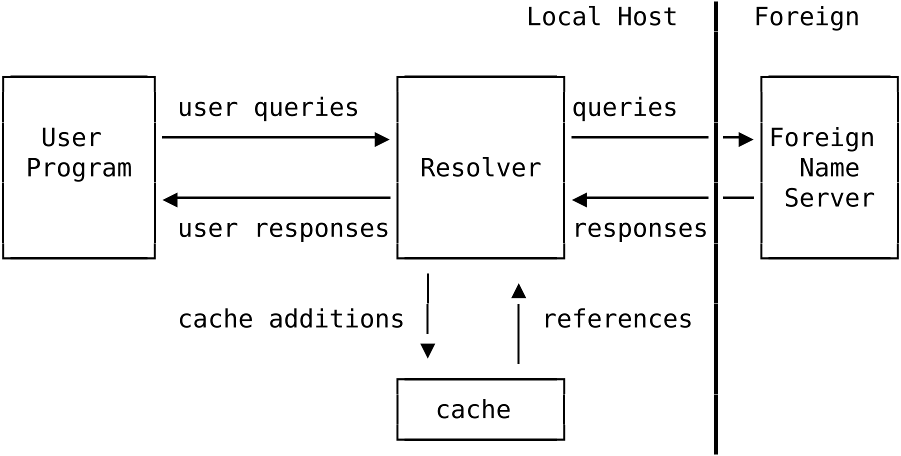
- A name (hostname) can be assigned to any device that has an IP address.
- The network software doesn’t require names, but they do make it easier for humans to use the network.

```
$ ssh username@cs3.swfu.edu.cn
```



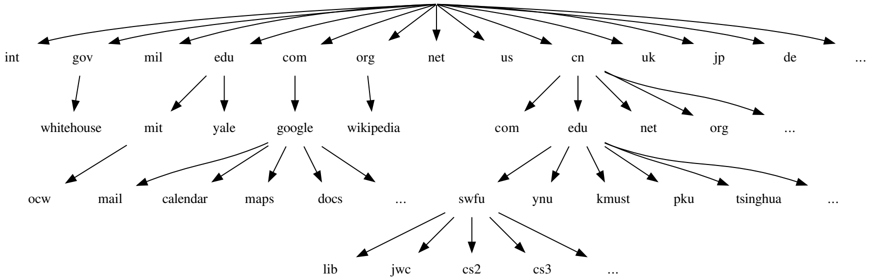
- Resolver is normally part of the application
  - `man 3 gethostbyname`
  - `man 3 gethostbyaddr`
- The TCP/IP protocols within the kernel know nothing about the DNS

# Typical Configuration



# The DNS Name Space Is Hierarchical

## The domain hierarchy is similar to the UNIX filesystem



- Organizational: `com`, `edu`, `gov`, `mil`, `net`, `org`, `int`
- Geographic: `cn`, `us`, `uk`, `jp`, `de`, etc.

# Translating Names Into Addresses

Two common ways:

Host table The old way. `/etc/hosts`

DNS A distributed database system — Domain Name Service (DNS)

# The Host Table

/etc/hosts

```
127.0.0.1          localhost
202.203.132.245    cs3.swfu.edu.cn    cs3
202.203.132.242    cs2.swfu.edu.cn    cs2
```

It’s still widely used, because:

- The important hosts on the local network
  - In case DNS is not running
- NIS host database
- Local intranet

# All hosts connected to the Internet should use DNS

## The old host table system is inadequate for the global Internet for two reasons:

1. inability to scale
2. lack of an automated update process.

## Old story

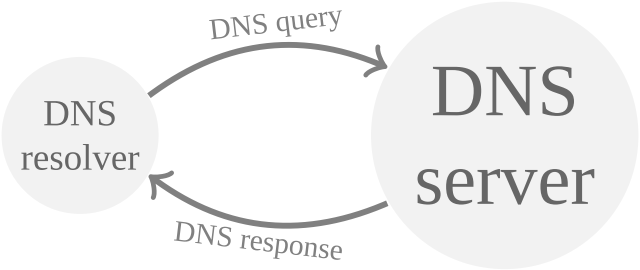
Prior to adopting DNS, the Network Information Center (NIC) maintained a large table of Internet hosts called the NIC host table. Hosts included in the table were called registered hosts, and the NIC placed hostnames and addresses into this file for all sites on the Internet.

# Domain Name System

- Scales well
  - Doesn’t rely on a single large table
  - Distributed database system that doesn’t bog down as the database grows

  DNS currently provides information on approximately 16,000,000 hosts, while less than 10,000 are listed in the host table.
- Guarantees that new host information will be disseminated to the rest of the network as it is needed

# DNS softwares

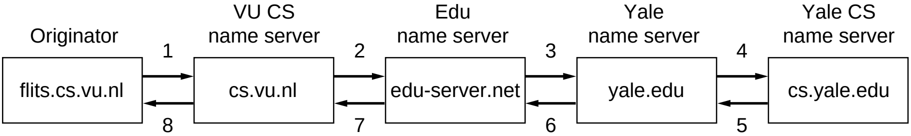


The resolver asks the questions.

The name server answers the questions.

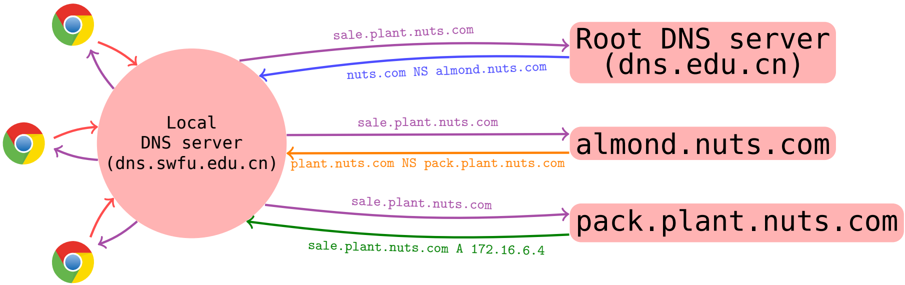
# Recursive Query

`flits.cs.vu.nl` wants to know the IP address of `linda.cs.yale.edu`



# Non-recursive Query

The remote server tells the local server who to ask next



# DNS Message Format

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr>
<tr><td colspan="16">ID</td></tr>
<tr><td>QR</td><td colspan="4">OPCODE</td><td>AA</td><td>TC</td><td>RD</td><td>RA</td><td>0</td><td>0</td><td>0</td><td colspan="4">RCODE</td></tr>
<tr><td colspan="16">QDCOUNT</td></tr>
<tr><td colspan="16">ANCOUNT</td></tr>
<tr><td colspan="16">NSCOUNT</td></tr>
<tr><td colspan="16">ARCOUNT</td></tr>
</table>

DNS Header

| DNS message |
|---|
| Header |
| Question |
| Answer |
| Authority |
| Additional |

## Flags:

QR: Query/Response
- 0: query
- 1: response

OPCODE: operation type
- 0 a standard query
- 1 an inverse query
- 2 server status request

AA: authoritative answer

TC: truncated. only the first 512 bytes of reply was returned

RD: Recursion Desired

RA: Recursion Available

RCODE: return code. common values:
- 0 no error
- 3 name error

```
~$ host -a cs2.swfu.edu.cn
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22237
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;cs2.swfu.edu.cn.       IN  ANY

;; ANSWER SECTION:
cs2.swfu.edu.cn.    3600    IN  A   202.203.132.242

Received 49 bytes from 114.114.114.114#53 in 1161 ms
```

# tcpdump

```
~$ host -a cs2.swfu.edu.cn
```

```
~$ sudo tcpdump -i wlan0 -n port 53
09:30:29.860901 IP 192.168.1.109.34075 > 114.114.115.115.53:
                34035+ ANY? cs2.swfu.edu.cn. (33)
09:30:29.979390 IP 114.114.115.115.53 > 192.168.1.109.34075:
                34035 1/0/0 A 202.203.132.242 (49)
```

| | | |
|---|---|---|
| `34035` | – | id |
| `+` | – | rd=1 |
| `ANY?` | – | query type |
| `33/49` | – | UDP payload length |
| `1/0/0` | – | **1** answer RR; **0** authority RR; **0** additional RR. |
| `A` | – | IPv4 address |

# Resource Records Example

```
~$ host -a mirrors.ustc.edu.cn
Trying "mirrors.ustc.edu.cn"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4421
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 2, ADDITIONAL: 4

;; QUESTION SECTION:
;mirrors.ustc.edu.cn.           IN      ANY

;; ANSWER SECTION:
mirrors.ustc.edu.cn.    600     IN      AAAA    2001:da8:d800:95::110
mirrors.ustc.edu.cn.    600     IN      A       202.38.95.110
mirrors.ustc.edu.cn.    594     IN      NS      f1g1ns2.dnspod.net.
mirrors.ustc.edu.cn.    594     IN      NS      f1g1ns1.dnspod.net.

;; AUTHORITY SECTION:
mirrors.ustc.edu.cn.    594     IN      NS      f1g1ns1.dnspod.net.
mirrors.ustc.edu.cn.    594     IN      NS      f1g1ns2.dnspod.net.

;; ADDITIONAL SECTION:
f1g1ns1.dnspod.net.     33536   IN      A       111.30.132.180
f1g1ns1.dnspod.net.     33536   IN      A       113.108.80.138
f1g1ns2.dnspod.net.     33536   IN      A       101.226.30.224
f1g1ns2.dnspod.net.     33536   IN      A       112.90.82.194

Received 323 bytes from 202.203.132.100#53 in 6598 ms
```

# Resource Records

## What's associated with a domain name?

| **Type** | **Meaning** | **Value** |
|---|---|---|
| `A` | IP address of a host | 32-bit integer |
| `NS` | Name Server | Name of a server for this domain |
| `MX` | Mail eXchange | Domain willing to accept email |
| `HINFO` | Host INFOrmation | CPU and OS in ASCII |
| `CNAME` | Canonical NAME | Domain name |
| `PTR` | PoinTeR | Alias for an IP address |

When a resolver gives a domain name to DNS, what it gets back are the *resource records* associated with that name.

# DNS References

Wikipedia. *Domain Name System — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Domain%5C_Name%5C_System&oldid=656963793.

MOCKAPETRIS P. *Domain names - concepts and facilities*. RFC Editor. 1987. https://www.rfc-editor.org/rfc/rfc1034.txt.

MOCKAPETRIS P. *Domain names - implementation and specification*. RFC Editor. 1987. https://www.rfc-editor.org/rfc/rfc1035.txt.

# E-mail Protocols

## Proprietary protocols:

Microsoft: Outlook client ⟺ Exchange server

IBM: Notes client ⟺ Domino server

## Open standards:

SMTP: Simple Mail Transfer Protocol, RFC2821

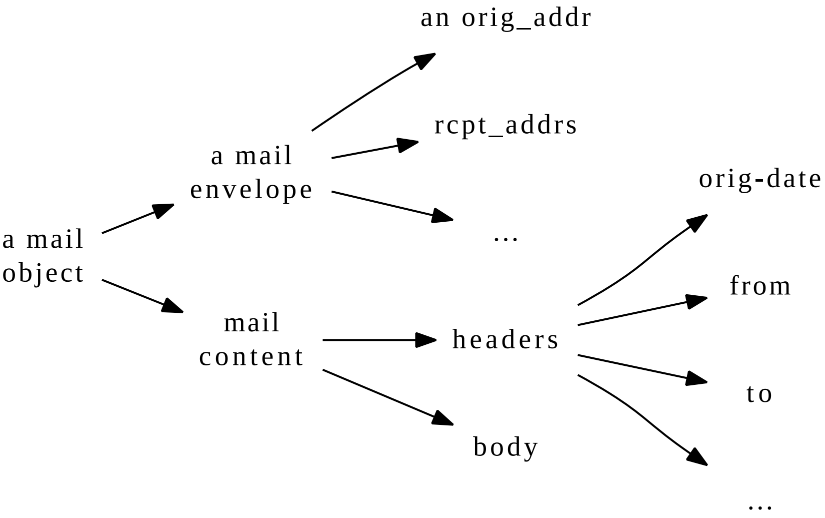
POP3: Post Office Protocol, RFC1939

MIME: Multipurpose Internet Mail Extensions, RFC2045, RFC2046, RFC2047, RFC2048, RFC2049

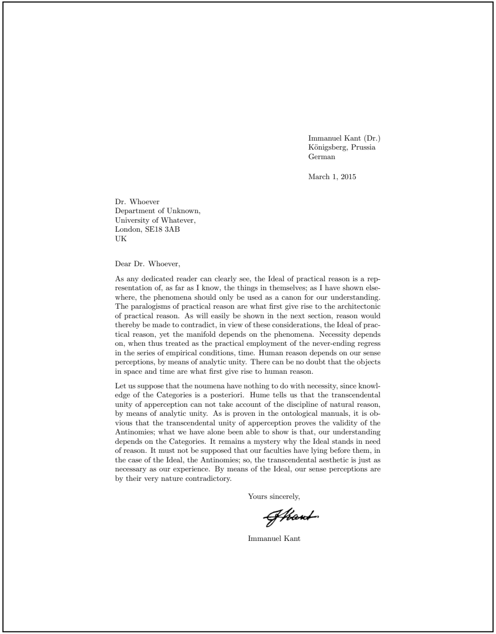
IMAP4: Interactive Mail Access Protocol, RFC3501

# SMTP Transports A Mail Object

## A Mail Object



# A Physical Mail



Immanuel Kant (Dr.)
Königsberg, Prussia
German

March 1, 2015

Dr. Whoever
Department of Unknown,
University of Whatever,
London, SE18 3AB
UK

Dear Dr. Whoever,

As any dedicated reader can clearly see, the Ideal of practical reason is a representation of, as far as I know, the things in themselves; as I have shown elsewhere, the phenomena should only be used as a canon for our understanding. The paralogisms of practical reason are what first give rise to the architectonic of practical reason. As will easily be shown in the next section, reason would thereby be made to contradict, in view of these considerations, the Ideal of practical reason, yet the manifold depends on the phenomena. Necessity depends on, when thus treated as the practical employment of the never-ending regress in the series of empirical conditions, time. Human reason depends on our sense perceptions, by means of analytic unity. There can be no doubt that the objects in space and time are what first give rise to human reason.

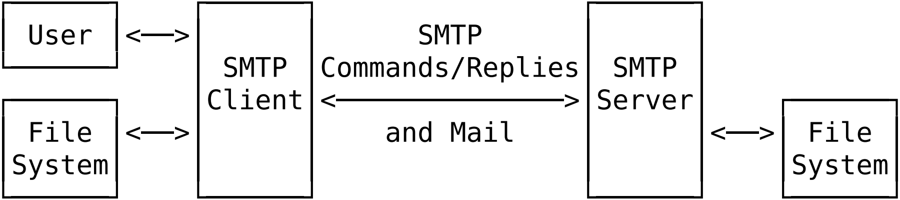
Let us suppose that the noumena have nothing to do with necessity, since knowledge of the Categories is a posteriori. Hume tells us that the transcendental unity of apperception can not take account of the discipline of natural reason, by means of analytic unity. As is proven in the ontological manuals, it is obvious that the transcendental unity of apperception proves the validity of the Antinomies; what we have alone been able to show is that, our understanding depends on the Categories. It remains a mystery why the Ideal stands in need of reason. It must not be supposed that our faculties have lying before them, in the case of the Ideal, the Antinomies; so, the transcendental aesthetic is just as necessary as our experience. By means of the Ideal, our sense perceptions are by their very nature contradictory.

Yours sincerely,

J Kant

Immanuel Kant

## The SMTP Basic Structure



User

File System

SMTP Client

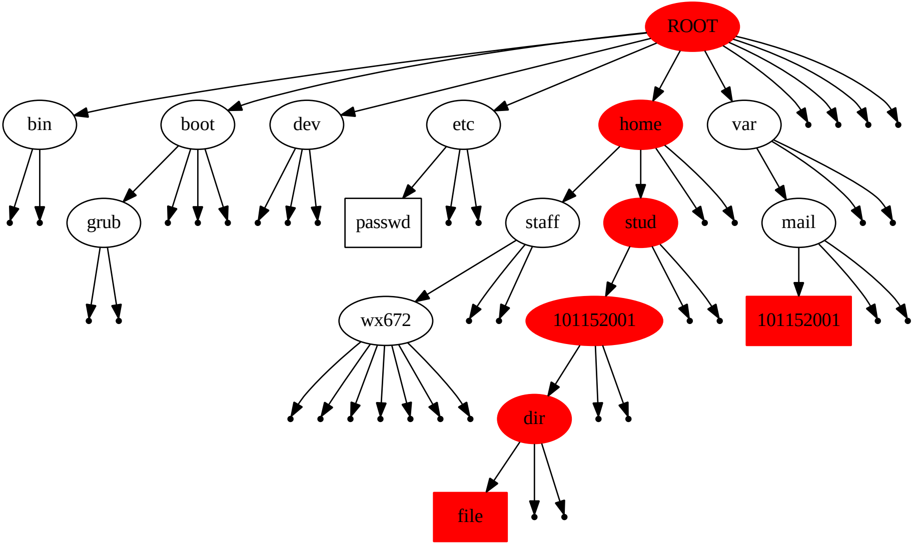
SMTP Commands/Replies and Mail

SMTP Server

File System

- TCP, port 25

# Unix File System



## SMTP Commands

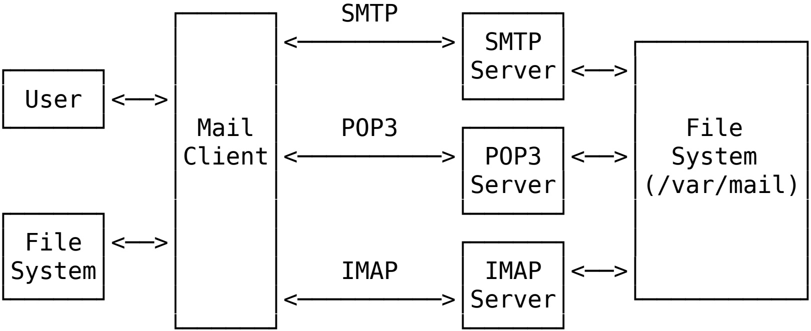
```
wx672@cs3:~$ nc localhost 25
220 cs3.swfu.edu.cn ESMTP Exim 4.72 Sun, 16 Oct 2011 22:29:29 +0800
help
214-Commands supported:
214 AUTH HELO EHLO MAIL RCPT DATA NOOP QUIT RSET HELP
```

- More commands can be available, depending on your SMTP server configuration.

# A Simple Protocol

## A SMTP Session

```
~$ nc cs3.swfc.edu.cn smtp
220 cs3.swfu.edu.cn ESMTP Exim 4.72
        Sun, 16 Oct 2011 22:18:22 +0800
helo debian
250 cs3.swfc.edu.cn Hello debian [192.168.128.5]
mail from:<wx672@debian>
250 OK
rcpt to:<wx672@cs3.swfc.edu.cn>
250 Accepted
data
354 Enter message, ending with "." on a line by itself
Hello, there!
.
250 OK id=1DMJra-0007IR-01
quit
221 cs3.swfc.edu.cn closing connection
wx672@debian:~$
```



User
File
System
Mail
Client
SMTP
POP3
IMAP
SMTP
Server
POP3
Server
IMAP
Server
File
System
(/var/mail)

# Post Office Protocol v3

| | |
|---|---|
| POP2 | port 109 |
| POP3 | port 110 |

The POP protocols verify the user's login name and password, and move the user's mail from the server to the user's local mail reader.

## A POP3 Session

```
~$ nc cs3 110
+OK Dovecot ready.
user wx672
+OK
pass topsecrete
+OK Logged in.
stat
+OK 3 459
retr 1
+OK 146 octets
  The full text of message 1
dele 1
+OK message # 1 deleted
retr 2
+OK 155 octets
  The full text of message 2
dele 2
+OK message # 2 deleted
retr 3
+OK 158 octets
  The full text of message 3
dele 3
+OK message # 3 deleted
quit
+OK Logging out.
```

# IMAP — Internet Message Access Protocol

- port 143

## Advantages over POP3

- Both connected and disconnected modes of operation
- Multiple clients can simultaneously connect to the same mailbox
- Access to MIME parts of messages and partial fetch
- Message state information kept on the server
- Multiple mailboxes on the server
- Server-side searches
- A built-in extension mechanism

# An IMAP session

```
~$ nc cs3 143
* OK Dovecot ready.
a001 login wx672 topsecrete
a001 OK Logged in.
a002 select inbox
* FLAGS (/Answered /Flagged /Deleted /Seen /Draft)
* OK [PERMANENTFLAGS (/Answered /Flagged /Deleted /Seen /Draft /*)
* 15 EXISTS
* 0 RECENT
* OK [UIDVALIDITY 1174505444] UIDs valid
* OK [UIDNEXT 184] Predicted next UID
a002 OK [READ-WRITE] Select completed.
a004 fetch 1 full
* 1 FETCH (FLAGS (/Seen) INTERNALDATE "16-Oct-2011 22:40:55 +0800"
a004 OK Fetch completed.
a006 fetch 1 body[text]
* 1 FETCH (BODY[TEXT] 55
hello ,there!
)
a006 OK Fetch completed.
a007 logout
* BYE Logging out
a007 OK Logout completed.
```

## Disadvantages of IMAP

- IMAP is a very heavy and complicated protocol
- IMAP generally results in higher server loads than POP3
- Server-side searches can potentially use lots of server resources when searching massive mailboxes

# Multipurpose Internet Mail Extensions

- SMTP supports only 7-bit ASCII characters.
- MIME standard defines mechanisms for emailing other kinds of information, e.g.
  - text in languages other than English,
  - files containing images, sounds, movies,
  - computer programs
- HTTP/MIME

# A Typical Mail Header

```
Received: from 20030704041 by cs2.swfc.edu.cn with local (Exim 4.50)
     id 1GSusu-0001D0-NT
     for wx672@cs2.swfc.edu.cn; Thu, 28 Sep 2006 20:21:00 +0800
Date: Thu, 28 Sep 2006 20:21:00 +0800
To: WANG Xiaolin <wx672@cs2.swfc.edu.cn>
Subject: ipv6
Message-ID: <20060928122100.GA4498@cs2.swfc.edu.cn>
Mime-Version: 1.0
Content-Type: text/plain; charset=utf-8
Content-Disposition: inline
Content-Transfer-Encoding: 8bit
User-Agent: Mutt/1.5.9i
From: 20030704041@cs2.swfc.edu.cn
X-SA-Exim-Connect-IP: <locally generated>
X-SA-Exim-Rcpt-To: wx672@cs2.swfc.edu.cn
X-SA-Exim-Mail-From: 20030704041@cs2.swfc.edu.cn
X-SA-Exim-Scanned: No (on cs2.swfc.edu.cn); SAEximRunCond expanded to false
X-Spam-Checker-Version: SpamAssassin 3.0.3 (2005-04-27) on cs2.swfc.edu.cn
X-Spam-Level: *
X-Spam-Status: No, score=1.0 required=5.0 tests=ALL_TRUSTED,AWL,FROM_ALL_NUMS,
     FROM_ENDS_IN_NUMS,FROM_STARTS_WITH_NUMS,NO_REAL_NAME autolearn=no
     version=3.0.3
Status: RO
Content-Length: 240
Lines: 3
X-UID: 351
X-Keywords:
```

# Spam

Spam:
- Any kind of un-wanted email messages.
- The action of sending such kinds of messages to usenet newsgroups, mailing lists, or any other individuals.

- by year 2000, 7% of Internet mails were spam;
- by year 2004, 60% were spam.
- Bill Gates receives nearly 4 million emails a day – most of which are spam.

# How Spam Works?

1. Collecting Email Addresses (Sniffing, Web Registration, Mailing List and Newsgroup, etc.)
2. Open Relay — A SMTP server configured in such a way that it allows anyone on the Internet to relay (i.e. send) email through it.
3. Open Proxy — A proxy which is misconfigured to allow access to anyone on the internet.

# Relayed Mail Scenario

```
wx672@cs2:~$ nc wx672.3322.org smtp
220 wx672.3322.org ESMTP Exim 4.50
        Tue, 03 Oct 2006 10:13:04 +0800
ehlo cs2.swfc.edu.cn
250-wx672.3322.org Hello cs2.swfc.edu.cn
        [202.203.132.242]
250-SIZE 52428800
250-PIPELINING
250 HELP
mail from:<wx672@cs2.swfc.edu.cn>
250 OK
rcpt to:<@wx672.3322.org:wx672@yahoo.com>
250 Accepted
data
354 Enter message, ending with "." on a line by itself
Hello, this is a message to wx672@yahoo.com
relayed by the smtp server at wx672.3322.org
.
250 OK id=1DSQRt-0000jC-T0
quit
221 wx672.3322.org closing connection
```

# Common Technologies Of Anti-Spams

- DNSBL — DNS-based Blackhole List
- Bayesian Filtering:

$$P(spam|words) = \frac{P(words|spam)P(spam)}{P(words)}$$

- Greylisting — ”normal” MTAs should attempt retries if given an appropriate temporary failure code for a delivery attempt.

# Mail References I

- Wikipedia. *Simple Mail Transfer Protocol — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Simple%5C_Mail%5C_Transfer%5C_Protocol&oldid=646541423.

- Wikipedia. *Post Office Protocol — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Post%5C_Office%5C_Protocol&oldid=645436521.

- Wikipedia. *Internet Message Access Protocol — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=Internet%5C_Message%5C_Access%5C_Protocol&oldid=647958613.

- Wikipedia. *MIME — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=MIME&oldid=644193928.

- KLENSIN (ED.) J. *Simple Mail Transfer Protocol*. RFC Editor. 2001. https://www.rfc-editor.org/rfc/rfc2821.txt.

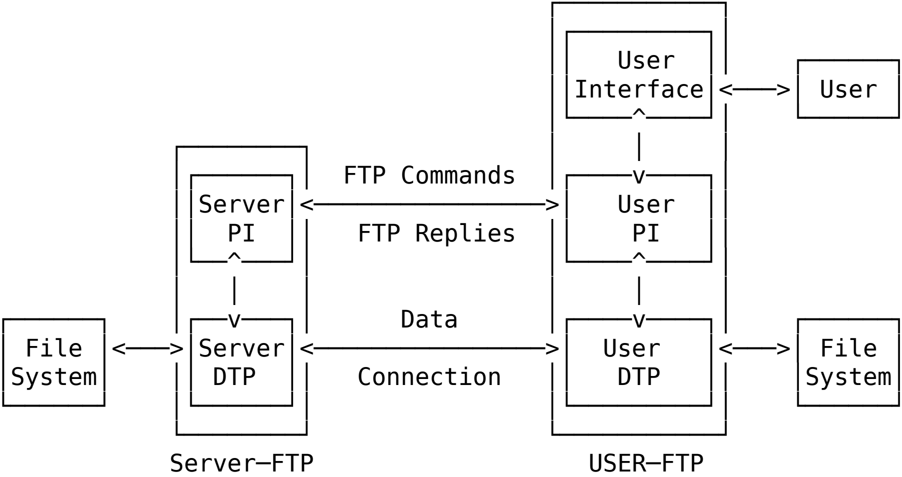
- MYERS J, ROSE M. *Post Office Protocol - Version 3*. RFC Editor. 1996. https://www.rfc-editor.org/rfc/rfc1939.txt.

# Mail References II

- CRISPIN M. *INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1*. RFC Editor. 2003. https://www.rfc-editor.org/rfc/rfc3501.txt.

- FREED N, BORENSTEIN N. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. RFC Editor. 1996. https://www.rfc-editor.org/rfc/rfc2045.txt.

# FTP


User Interface
User
FTP Commands
Server PI
User PI
FTP Replies
Data
File System
Server DTP
User DTP
File System
Connection
Server-FTP
USER-FTP

# An Active FTP Session

## Control session

| Server | Client |
|---|---|
| cs2 ⇒ 202.203.132.242 | cs3 ⇒ 202.203.132.245 |

```
wx672@cs3:~$ nc cs2 ftp
220 (vsFTPd 2.0.5)
user wx672
331 Please specify the password.
pass canttellyou
230 Login successful.
port 202,203,132,245,100,0
200 PORT command successful. Consider using PASV.
nlst
150 Here comes the directory listing.
226 Directory send OK.
quit
221 Goodbye.
```

```
port 202,203,132,245,100,0
     └──────┬──────┘ └───┴─> Port (2 x 8 bits)
            └──────────────>   IP (4 x 8 bits)
```

## To see FTP data session:

```
wx672@cs3:~$  nc -l $((100*256+0))
```

# A Passive FTP Session

## Control session

| Server | Client |
|---|---|
| cs2 ⇒ 202.203.132.242 | cs3 ⇒ 202.203.132.245 |

```
wx672@cs3:~$ nc cs2 ftp
220 (vsFTPd 2.0.5)
user wx672
331 Please specify the password.
pass canttellyou
230 Login successful.
pasv
227 Entering Passive Mode (202,203,132,242,36,5)
list
150 Here comes the directory listing.
quit
221 Goodbye.
```

## To see FTP data session:

```
wx672@cs3:~$ nc cs2 $((36*256+5))
```

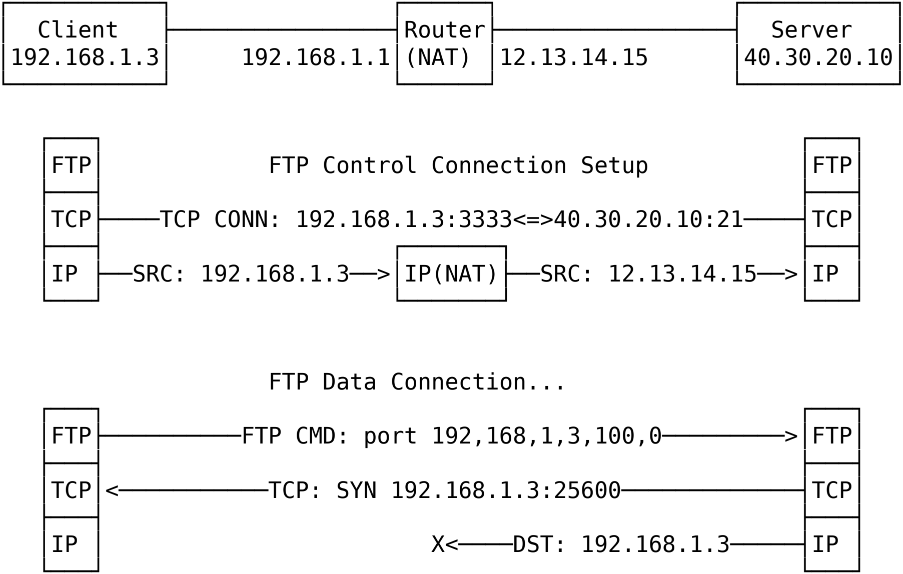
# Active FTP vs. Passive FTP

In active mode: Server initiates data connection to client’s data port.

In passive mode: Client initiates data connection to random port specified by server.

# Why Passive Mode?

## Active mode doesn’t work with firewall



Client
192.168.1.3

192.168.1.1

Router
(NAT)

12.13.14.15

Server
40.30.20.10

FTP Control Connection Setup

FTP | TCP | IP

TCP CONN: 192.168.1.3:3333<=>40.30.20.10:21

SRC: 192.168.1.3—>

IP(NAT)

SRC: 12.13.14.15—>

FTP | TCP | IP

FTP Data Connection...

FTP | TCP | IP

FTP CMD: port 192,168,1,3,100,0—>

<—TCP: SYN 192.168.1.3:25600

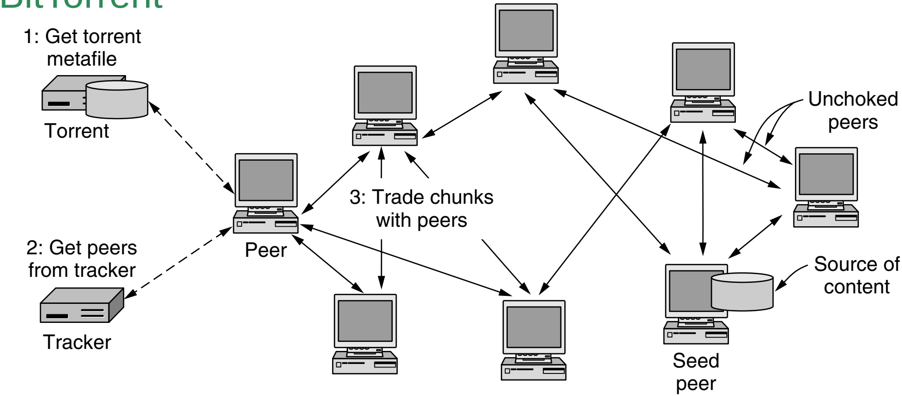
X<—DST: 192.168.1.3

FTP | TCP | IP

# FTP References

- Wikipedia. *File Transfer Protocol — Wikipedia, The Free Encyclopedia*. 2015. http://en.wikipedia.org/w/index.php?title=File%5C_Transfer%5C_Protocol&oldid=647883278.
- POSTEL J, REYNOLDS J. *File Transfer Protocol*. RFC Editor. 1985. https://www.rfc-editor.org/rfc/rfc959.txt.
- BELLOVIN S. *Firewall-Friendly FTP*. RFC Editor. 1994. https://www.rfc-editor.org/rfc/rfc1579.txt.

# BitTorrent



1. How does a peer find other peers that have the content it wants to download?
2. How is content replicated by peers to provide high-speed downloads for everyone?
3. How do peers encourage each other to upload content to others as well as download content for themselves?

# P2P References

Wikipedia. *BitTorrent — Wikipedia, The Free Encyclopedia*. 2015.
http://en.wikipedia.org/w/index.php?title=BitTorrent&oldid=648034964.

COHEN B. *The BitTorrent Protocol Specification, Version 11031*. 2008.
http://www.bittorrent.org/beps/bep%5C_0003.html.