# Network Basics

Lecture Handouts

Wang Xiaolin

wx672ster+net@gmail.com

May 29, 2019

## Contents

[1] Wikipedia. Computer network — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Computer_network&oldid=647677143.

[2] ZAKON R. Hobbes' Internet Timeline. RFC Editor. 1997. https://www.rfc-editor.org/rfc/rfc2235.txt.

[3] Wikipedia. Google — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Google&oldid=649655452.

[4] Wikipedia. Internet protocol suite — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Internet_protocol_suite&oldid=647730161.

[5] Wikipedia. Don't be evil — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Don%27t_be_evil&oldid=650206338.

[6] Wikipedia. List of Google products — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=List_of_Google_products&oldid=649748144.

[7] Wikipedia. Internet Explorer — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Internet_Explorer&oldid=649893090.

[8] Wikipedia. Baidu — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Baidu&oldid=650739662.

[9] Wikipedia. Malware — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Malware&oldid=650004623.

[10] Wikipedia. Computer virus — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Computer_virus&oldid=650763583.

[11] Wikipedia. Adware — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Adware&oldid=648275939.

[12] Wikipedia. Spyware — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Spyware&oldid=650787662.

[13] Wikipedia. Computer worm — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Computer_worm&oldid=649547486.

[14] Wikipedia. Trojan horse (computing) — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Trojan_horse_(computing)&oldid=650679154.

[15] Wikipedia. Network topology — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Network_topology&oldid=650723398.

[16] Wikipedia. Network architecture — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Network_architecture&oldid=646400753.

[17] SOCOLOFSKY T, KALE C. TCP/IP tutorial. RFC Editor. 1991. https://www.rfc-editor.org/rfc/rfc1180.txt.

[18] TANENBAUM A, WETHERALL D. Computer Networks. Pearson Prentice Hall, 2011.

[19] KUROSE J, ROSS K. Computer Networking: A Top-down Approach. Pearson, 2013.

[20] BRADEN (ED.) R. Requirements for Internet Hosts - Communication Layers. RFC Editor. 1989. https://www.rfc-editor.org/rfc/rfc1122.txt.

[21] REYNOLDS J, POSTEL J. Assigned numbers. RFC Editor. 1985. https://www.rfc-editor.org/rfc/rfc943.txt.

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

[23] Wikipedia. Network address translation — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Network_address_translation&oldid=652836698.

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

[25] TYSON J. How LAN Switches Work — HowStuffWorks.com. 2001. %5Curl%7Bhttp://computer.howstuffworks.com/lan-switch.htm%7D.

[26] Cisco. Transparent Bridging — Cisco DocWiki. 2012. %5Curl%7Bhttp://docwiki.cisco.com/w/index.php?title=Transparent_Bridging&oldid=49091%7D.

[27] Wikipedia. Spanning Tree Protocol — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Spanning_Tree_Protocol&oldid=651745418.

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

[29] Wikipedia. Transmission Control Protocol — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Transmission_Control_Protocol&oldid=647944260.

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

[31] JACOBSON V, BRADEN R, BORMAN D. TCP Extensions for High Performance. RFC Editor. 1992. https://www.rfc-editor.org/rfc/rfc1323.txt.

[32] MATHIS M, MAHDAVI J, FLOYD S, et al. TCP Selective Acknowledgment Options. RFC Editor. 1996. https://www.rfc-editor.org/rfc/rfc2018.txt.

[33] POSTEL J. The TCP Maximum Segment Size and Related Topics. RFC Editor. 1983. https://www.rfc-editor.org/rfc/rfc879.txt.

[34] BORMAN D. TCP Options and Maximum Segment Size (MSS). RFC Editor. 2012. https://www.rfc-editor.org/rfc/rfc6691.txt.

[35] HARRENSTIEN K, STAHL M, FEINLER E. DoD Internet host table specification. RFC Editor. 1985. https://www.rfc-editor.org/rfc/rfc952.txt.

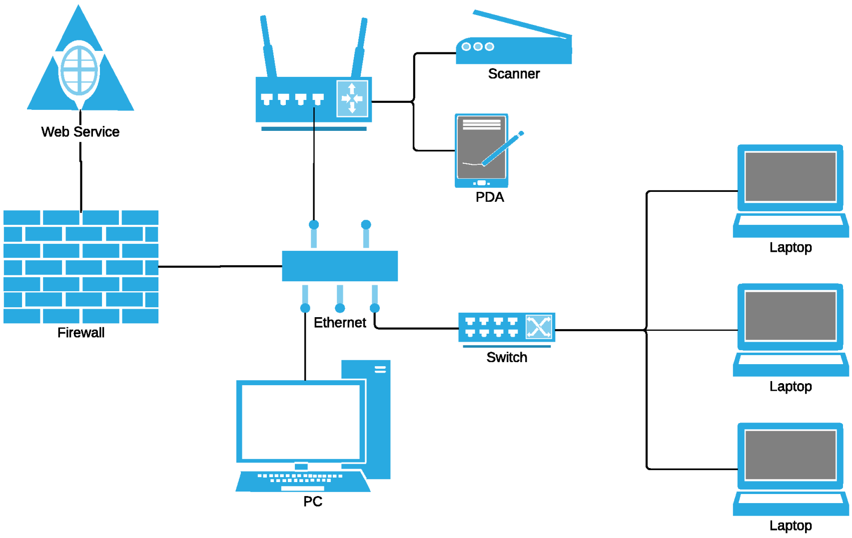
[36] FALL K, STEVENS W. TCP/IP Illustrated, Volume 1: The Protocols. Pearson Education, 2011.

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

# 1 Introduction

## 1.1 What's A Computer Network?

### What's A Computer Network?




See also: [*Computer network — Wikipedia, The Free Encyclopedia*]

## 1.2 Past and Future

### The History of Internet

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
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
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?

See also: [*Hobbes' Internet Timeline*]

## 1.3 The Internet

**What's The Internet?**

What pops up in your mind if I say "Internet"?

**For me, the answer is...**

Google

and...

`TCP/IP`

See also: [*Google — Wikipedia, The Free Encyclopedia*], [*Internet protocol suite — Wikipedia, The Free Encyclopedia*]

**What's The Internet?**

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

### 1.3.1 Google Philosophy

**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.

See also: [*Don't be evil — Wikipedia, The Free Encyclopedia*]

**Google Philosophy**

*More about...*

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

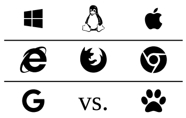
### 1.3.2 Google Products

**Google Products**

See also [*List of Google products — Wikipedia, The Free Encyclopedia*]

### 1.3.3 Choosing The Right Tools

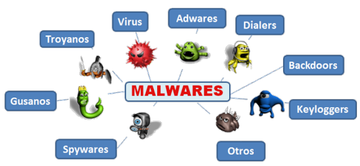
**Choosing The Right Tools**



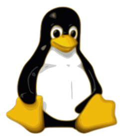

See also: [*Internet Explorer — Wikipedia, The Free Encyclopedia*], [*Baidu — Wikipedia, The Free Encyclopedia*]

### 1.3.4 Safe Surfing

**Dangerous**




**My solution**



See also: [*Malware — Wikipedia, The Free Encyclopedia*], [*Computer virus — Wikipedia, The Free Encyclopedia*], [*Adware — Wikipedia, The Free Encyclopedia*], [*Spyware — Wikipedia, The Free Encyclopedia*], [*Computer worm — Wikipedia, The Free Encyclopedia*], [*Trojan horse (computing) — Wikipedia, The Free Encyclopedia*]

**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

# 2 How The Internet Works

## 2.1 Network Classification

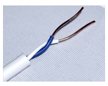
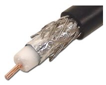
**Network Classification**

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

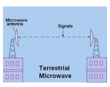
**Network Classification**

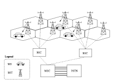
**Connection method**

Wired:


Wireless:

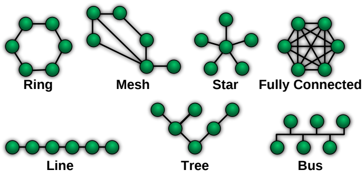



**Scale**

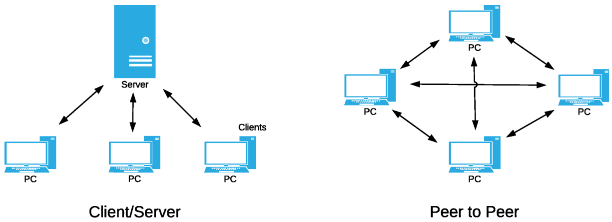
PAN, LAN, CAN, MAN, WAN ...

**Topology**




See also: [*Network topology — Wikipedia, The Free Encyclopedia*]

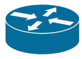
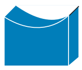
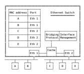
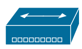
**Network Architecture**




See also: [*Network architecture — Wikipedia, The Free Encyclopedia*]

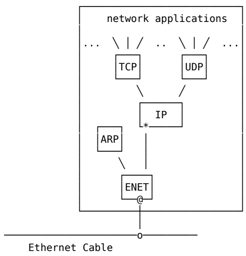
## 2.2 Basic Hardware Components

**Basic Hardware Components**

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

## 2.3 Network Architecture

**TCP/IP**



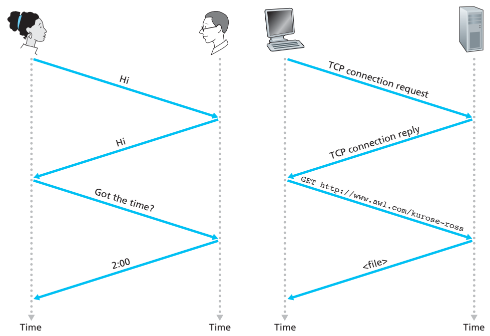

See also: [*Internet protocol suite — Wikipedia, The Free Encyclopedia*], [*TCP/IP tutorial*]

Each Internet-enabled computer has a TCP/IP protocol stack inside. And as you can see, the incoming packets will face a 1-to-N situation. The value of the *type* field in the Ethernet frame determines whether the Ethernet frame is passed to the ARP or the IP module. The *protocol* field in the IP header, and the port number in TCP/UDP header serve in a similar way.

### What's TCP/IP?

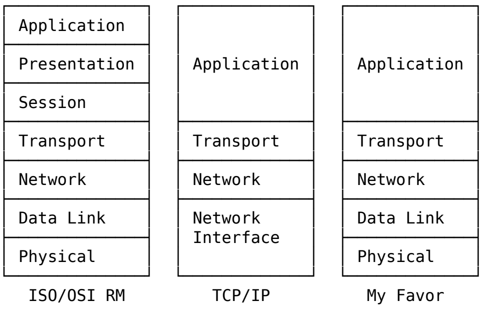
#### A set of protocols designed for the Internet

Protocol — a rule, a treaty, an agreement ...




### TCP/IP Protocol Stack

#### Every networked computer has it inside

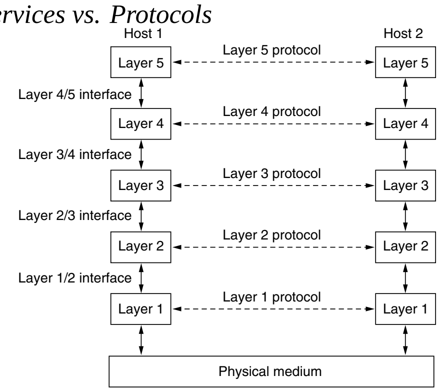


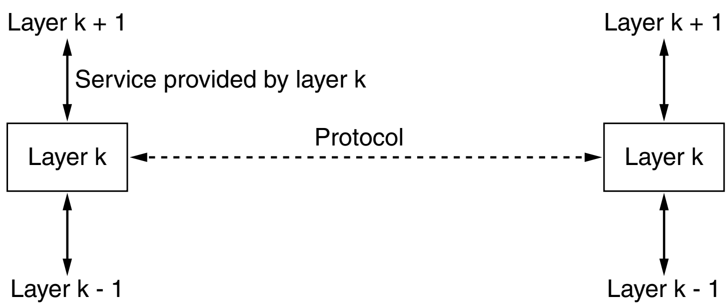

See also: [*Computer Networks*, Sec. 1.3, *Network Software*].

TCP/IP means everything related to TCP and IP.

### Layered Design

*Services vs. Protocols*




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

See also: [*Computer Networks*, Sec. 1.3.5, *The Relationship of Services to Protocols*]

**Network architecture**

**Architecture:** A big blueprint without worrying about any design details.

- A set of layers and protocols
- Neither the details of the implementation nor the specification of the interfaces is part of the architecture.

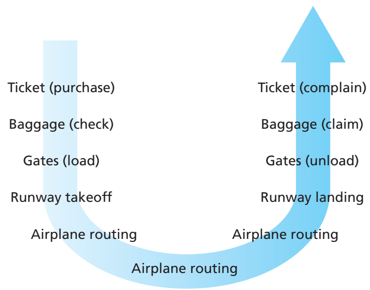
**Services** Interfaces between layers (primitives, functions)

**Protocols**

- for peer to peer talking
- The *internal implementation* of services provided by a layer
- It's common that different hosts use different implementations of the same protocols (e.g. Linux vs. Windows)
- Protocol changes have no effects on it's upper/lower layers

**Layered Design Example**

**Taking an airplane trip**




Each layer

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

See also:

- [*Computer Networking: A Top-down Approach*, Sec. 1.5.1, *Layered Architecture*]
- [*Computer Networks*, Sec. 1.3.2, *Design Issues for the Layers*]

**Layered design**

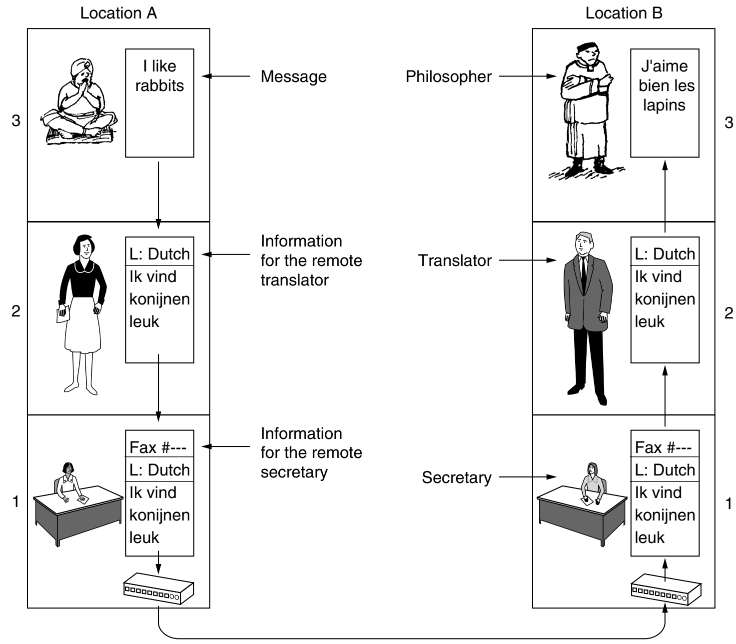
- Reduce design complexity
- Serve as a black box to its upper layer
  - Black box examples: information hiding, abstract data type, data encapsulation, object oriented programming

**Design issues**

- Reliability
  - never lost data — by means of acknowledgement
  - Error detection/correction
  - Routing
- Evolution
  - Protocol layering
  - Addressing
  - Internetworking: disassembling, transmitting, reassembling
  - Scalability
- Resource allocation

  - Statistical multiplexing
  - Flow control
  - Congestion
  - QoS
- Security
  - Authentication
  - Integrety

See also [*Requirements for Internet Hosts - Communication Layers*].




**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

An analogy may help explain the idea of multilayer communication. Imagine two philosophers (peer processes in layer 3), one of whom speaks Urdu and English and one of whom speaks Chinese and French. Since they have no common language, they each engage a translator (peer processes at layer 2), each of whom in turn contacts a secretary (peer processes in layer 1). Philosopher 1 wishes to convey his affection for oryctolagus cuniculus to his peer. To do so, he passes a message (in English) across the 2/3 interface to his translator, saying “I like rabbits,” as illustrated in Fig. 26. The translators have agreed on a neutral language known to both of them, Dutch, so the message is converted to “Ik vind konijnen leuk.” The choice of the language is the layer 2 protocol and is up to the layer 2 peer processes.[*Computer Networks*, Sec. 1.3, *Network Software*, p. 31]

The translator then gives the message to a secretary for transmission, for example, by email (the layer 1 protocol). When the message arrives at the other secretary, it is passed to the local translator, who translates it into French and passes it across the 2/3 interface to the second philosopher. Note that each protocol is completely independent of the other ones as long as the interfaces are not changed. The translators can switch from Dutch to, say, Finnish, at will, provided that they both agree and neither changes his interface with either layer 1 or layer 3. Similarly, the secretaries can switch from email to telephone without disturbing (or even informing) the other layers. Each process may add some information intended only for its peer. This information is not passed up to the layer above.

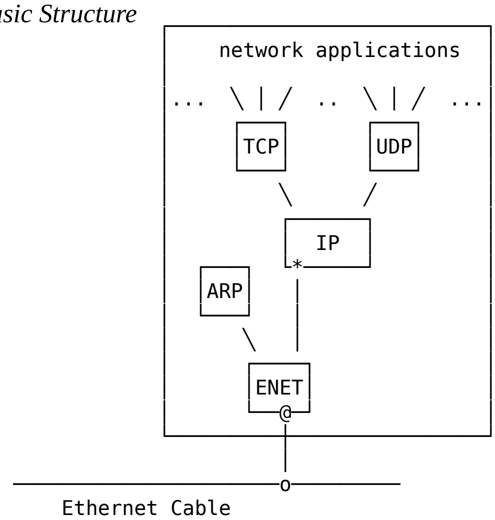
## 2.4 TCP/IP Overview

**Historical** The ARPANET was a research network sponsored by the DoD (U.S. Department of Defense). ... When satellite and radio networks were added later, the existing protocols had trouble interworking with them, so a new reference architecture was needed. Thus, *from nearly the beginning, the ability to connect multiple networks in a seamless way was one of the major design goals*. This architecture later became known as the TCP/IP Reference Model, after its two primary protocols. [*Computer Networks*, Sec. 1.4.2, *The TCP/IP Reference Model*]

Given the DoD's worry that some of its precious hosts, routers, and internetwork gateways might get blown to pieces at a moment's notice by an attack from the Soviet Union, another major goal was that the network be able to survive loss of subnet hardware, without existing conversations being broken off. In other words, *the DoD wanted connections to remain intact as long as the source and destination machines were functioning, even if some of the machines or transmission lines in between were suddenly put out of operation*. Furthermore, since applications with divergent requirements were envisioned, ranging from transferring files to real-time speech transmission, a flexible architecture was needed.

### `TCP/IP` Overview

*Basic Structure*



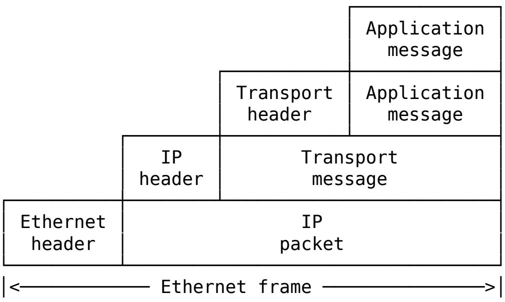

1. Where will an incoming Ethernet frame go?
   `0x0806` → ARP
   `0x0800` → IP
2. Where will an incoming IP packet go?
   `0x06` → TCP
   `0x11` → UDP
3. Where will an incoming transport message (UDP datagram, TCP segment) go?

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

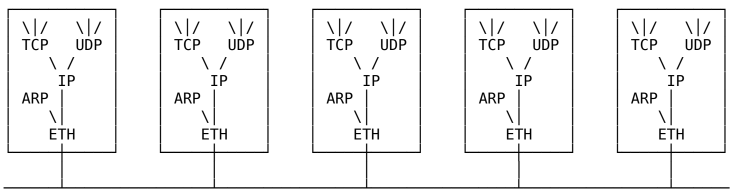
## 2.5 Terminology

**The Name Of A Unit Of Data**

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




## 2.6 Ethernet




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 |

**CSMA/CD** CSMA/CD is the communication protocol used by devices in the same Ethernet when they talk to each other.

- *Carrier Sense* means every device in the Ethernet can detect whether the carrier is busy. Carrier, the media used for carrying electric signals, is the Ethernet cable;
- *Multiple Access* means every device has equal chance of using the carrier for data transmission ;
- *Collision Detection* means if more than one devices trying to transmit data at the same time, they can detect this collision, and wait for a random (but short) period before trying to transmit again.

See [*TCP/IP tutorial*, Sec. 3.1] for a human analogy.

### Ethernet References


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

[2] Wikipedia. Ethernet frame — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Ethernet_frame&oldid=653337888.

[3] Wikipedia. Carrier sense multiple access with collision detection — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Carrier_sense_multiple_access_with_collision_detection&oldid=648771859.

[4] 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.

## 2.7 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**

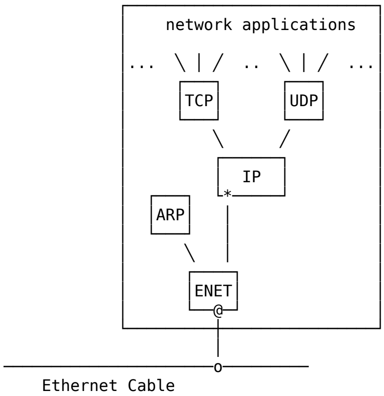
| IP address | Ethernet address |
|---|---|
| 223.1.2.1 | 08-00-39-00-2F-C3 |
| 223.1.2.2 | 08-00-28-00-38-A9 |
| 223.1.2.3 | 08-00-5A-21-A7-22 |
| 223.1.2.4 | 08-00-10-99-AC-54 |

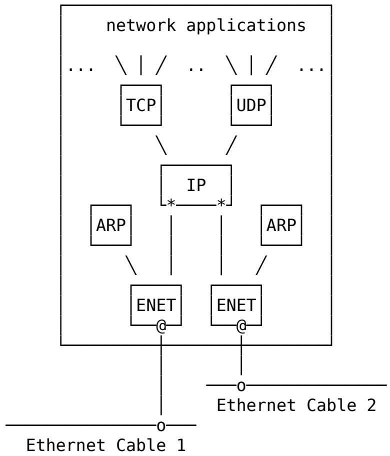
**ARP References**


[1] Wikipedia. Address Resolution Protocol — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Address_Resolution_Protocol&oldid=647148843.

[2] 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.

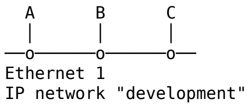
## 2.8 IP




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

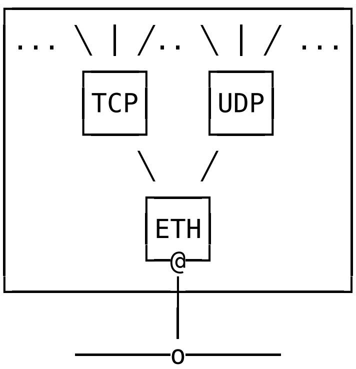
### Direct Routing—IP is overhead

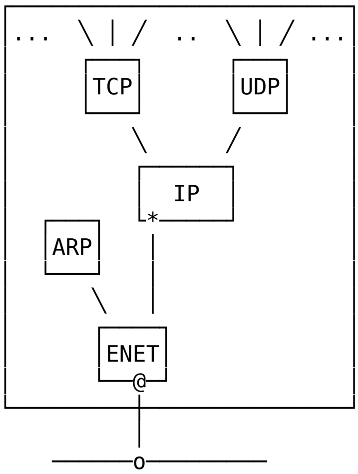



### Addresses in an Ethernet frame for an IP packet from A to B

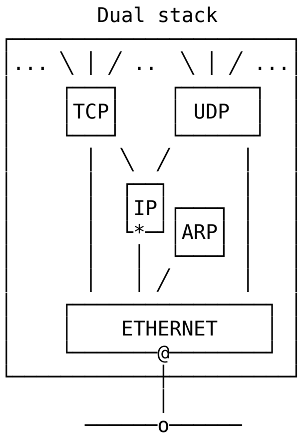
| address | source | destination |
|---|---|---|
| IP | A | B |
| Eth | A | B |

### Is IP Necessary?

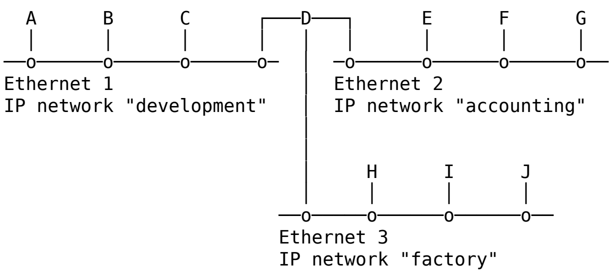



[1em]




### Indirect Routing



**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)**

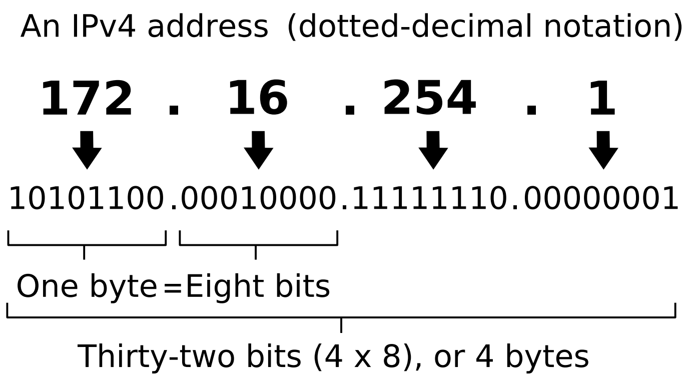
| address | source | destination |
|---|---|---|
| IP | A | E |
| Eth | D | E |

**IP Module Routing Rules**

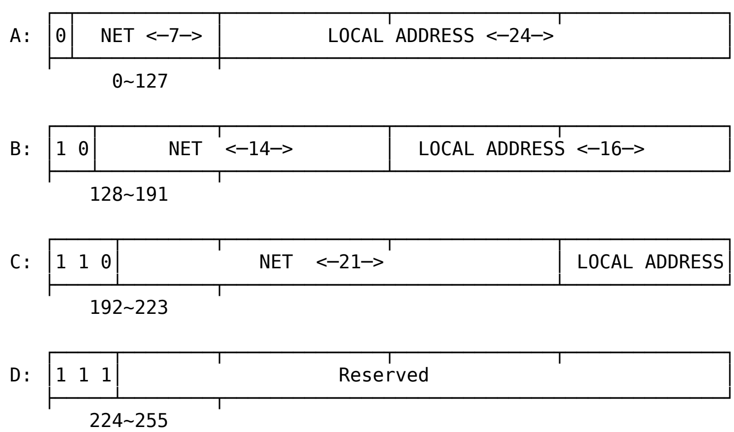
1. For an outgoing IP packet, entering IP from an upper layer, IP must decide
   - whether to send the IP packet directly or indirectly, and
   - IP must choose a lower network interface.

   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.

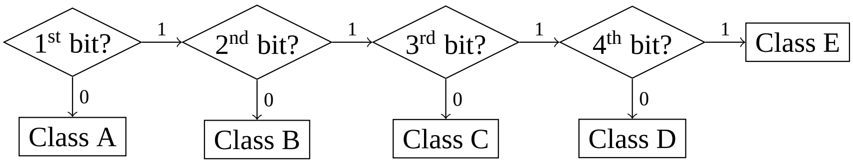
**IP Address**




**Address classes**



**Prefix**




**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.

See also: [*Assigned numbers*, *RFC 943*]

**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
```

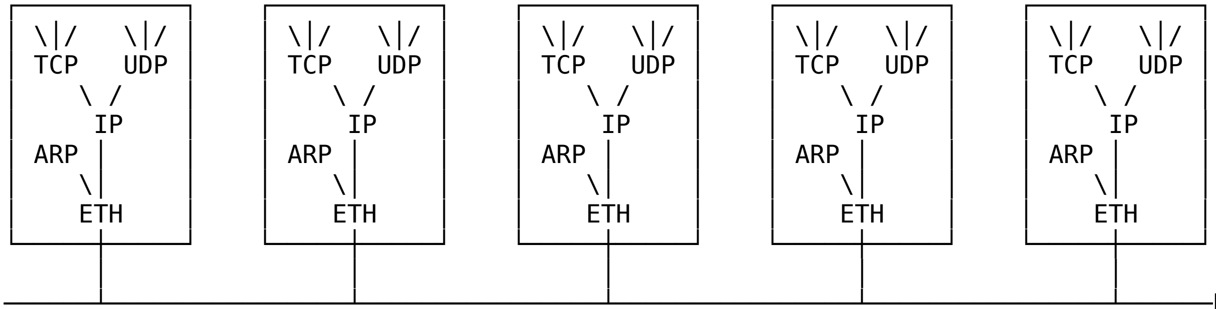
**Direct Routing Details**

```
  223.1.2.1         223.1.2.2
  ┌─────────┐       ┌─────────┐
  │ alpha   │       │ beta    │
  │     1   │       │  1      │
  └─────@───┘       └──@──────┘
        |              |
  ──────o──────────────o─────
Ethernet 1
IP network "development"
            223.1.2.0
```

**The route table inside alpha (simplified)**

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

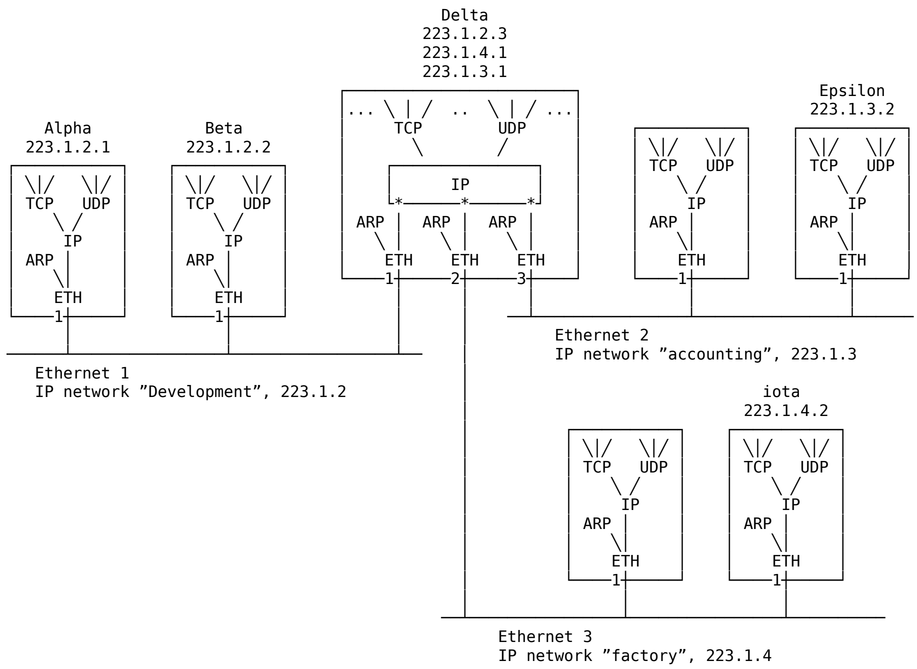
### Homework




[2em]

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

### Indirect Routing Details




### 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 |

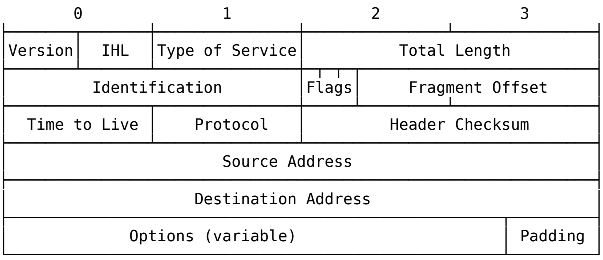
### The route table inside delta

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

### Managing The Routes

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

### IP Packet




### IP References


[1] Wikipedia. Internet Protocol — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Internet_Protocol&oldid=645719875.

[2] Wikipedia. IP address — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=IP_address&oldid=651153507.

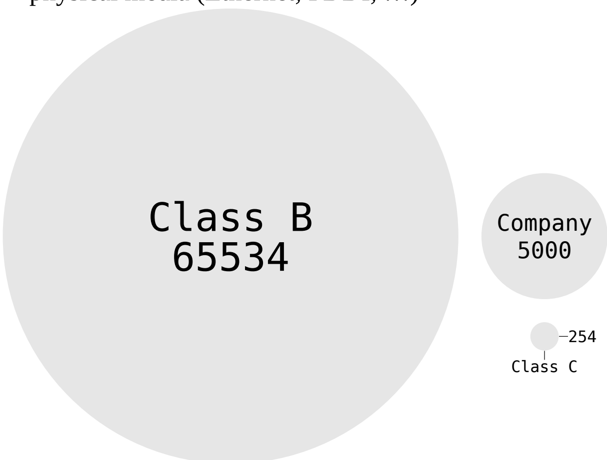
[3] POSTEL J. Internet Protocol. RFC Editor. 1981. https://www.rfc-editor.org/rfc/rfc791.txt.

[4] Wikipedia. IPv4 header checksum — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=IPv4_header_checksum&oldid=645516564.

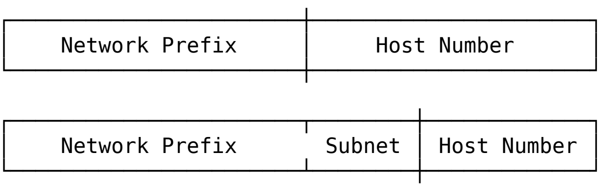

## 2.9 Subnetting

### Why Subnetting?

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




### How

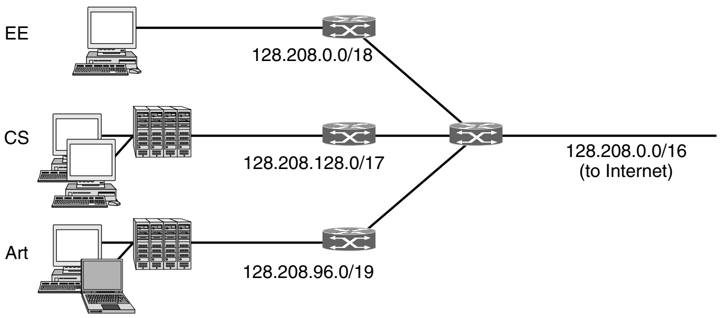


**Subnet mask** is a bitmask used to identify the network and node parts of the address.

**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
```

See detailed explanation in [*Computer Networks*, P.445].

**$2^n - 2$**

**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)

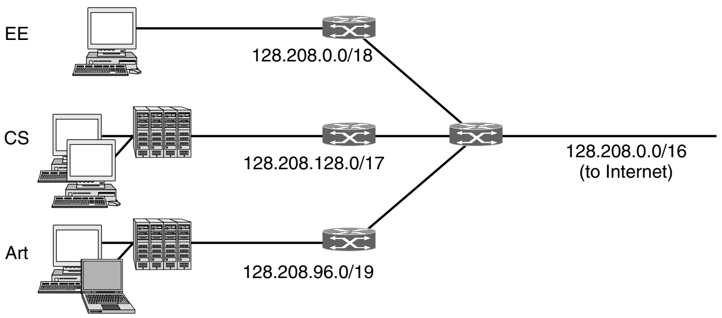
**Subnet Calculator**

**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**


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

[2] Wikipedia. IPv4 subnetting reference — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=IPv4_subnetting_reference&oldid=652583338.

[3] Wikipedia. Private network — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Private_network&oldid=649931709.

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

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


## 2.10 CIDR

**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:

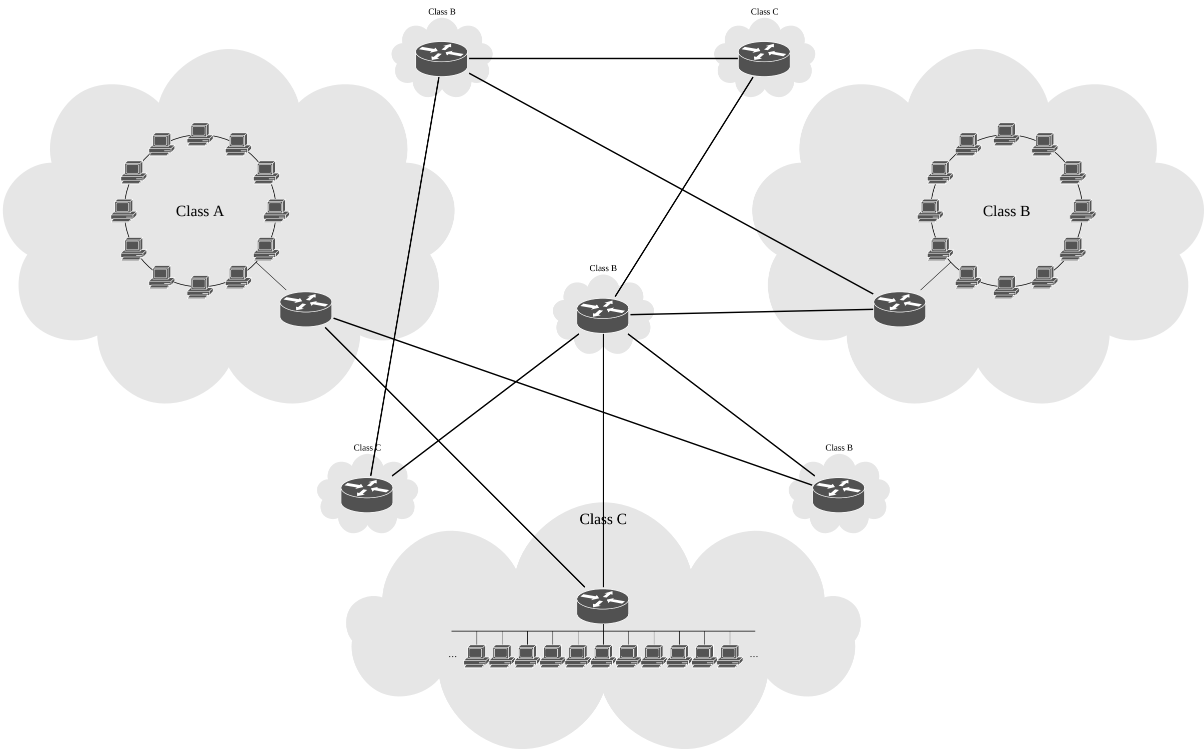
- Running out of IP addresses
- Running out of capacity in the global routing tables

From https://searchnetworking.techtarget.com/definition/CIDR:

> CIDR is a more flexible way to allocate IP addresses than the original scheme. As a result, the number of IP addresses was greatly increased, which along with widespread use of network address translation (NAT), has significantly extended the useful life of IPv4.

**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.

The old classful scheme was very inefficient in IP address allocation especially in class B. As we can imagine, for an medium-sized business, class A is too large, while class C is too small. So usually class B is the choice though it still is far too large for most organizations. In reality, more than half of class B networks have fewer than 50 hosts. Obviously, a class C network would have done the job, but everyone that asked for a class B network thought his business would outgrow the 8-bit host field.

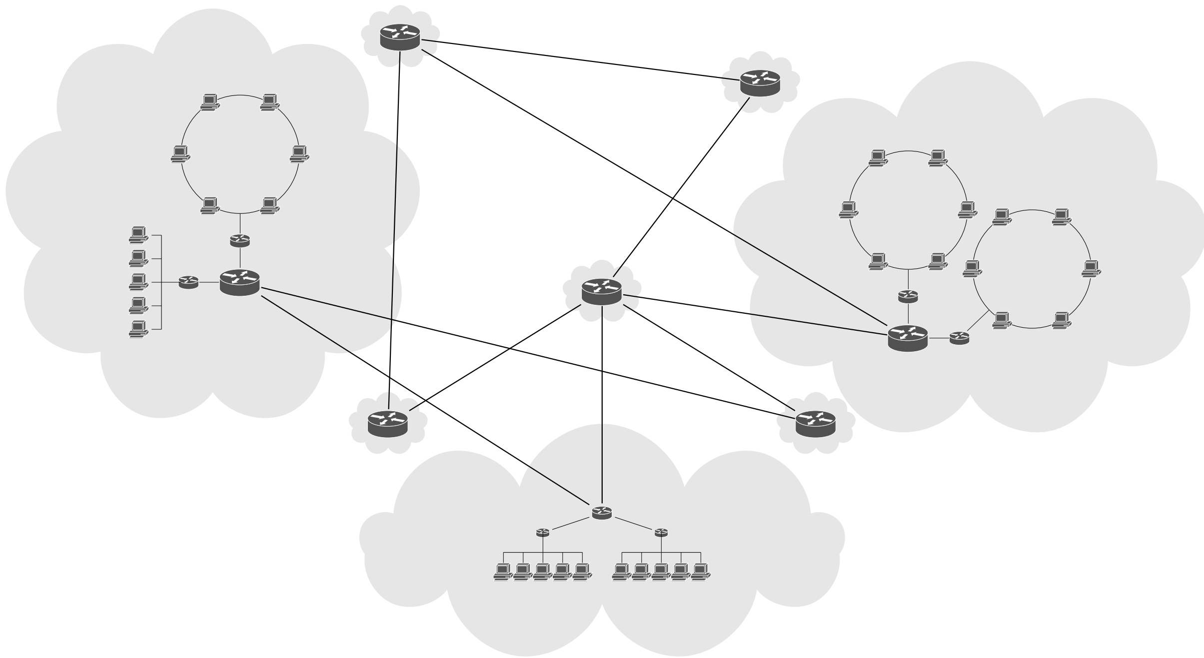
To handle this problem, subnets were introduced to flexibly assign blocks of addresses within an organization.

**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.

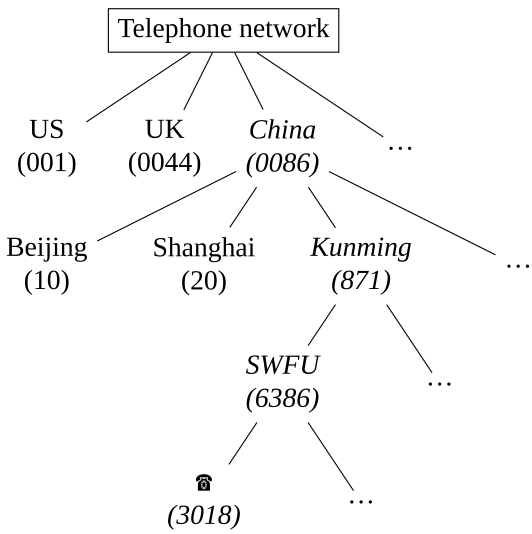
Subnetting can significantly improve the efficiency of address assignments. But it can also increase the size of the routing table. For example, a routing table could have a class B network listed in it. But after subnetting, this class B network has been split into 10 subnets. As a result, in the routing table the one line class B network record has been replaced by 10 lines of subnets.

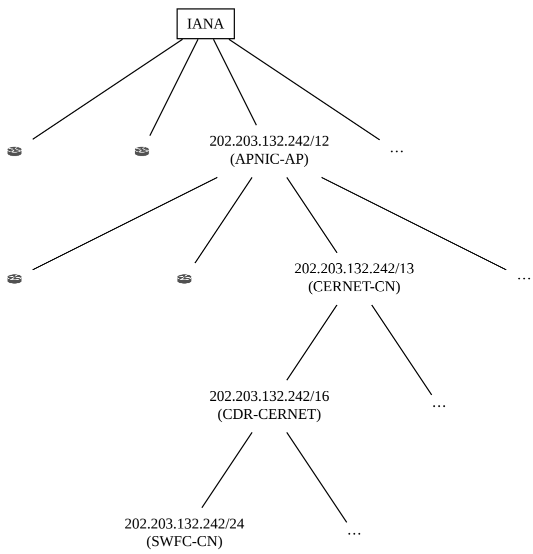
**Multi-Level Internet**



**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.



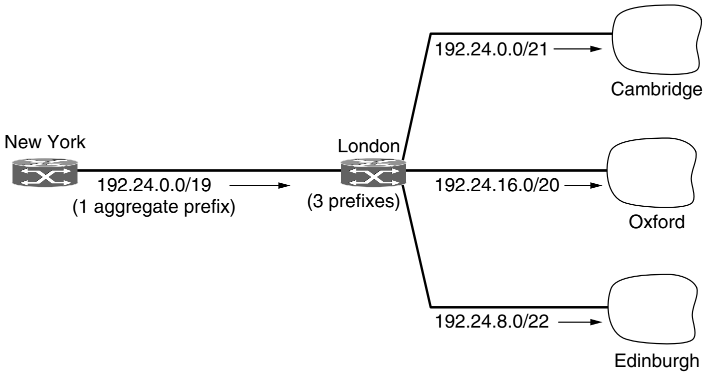




**Restructuring IP Address Assignments**

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 (= 1 Class B) | 65,536 hosts |
| /13 | 2,408 Class C | 524,288 hosts |

**Example**




**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 |

In this example, a block of addresses (192.24.0.0 ~192.24.31.255) in a class B network has been split into 3 subnets. The routing table inside the router in London should look like this:

| 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 |
| … | … | … | … |

While the routing table inside the router in New York...

**The Router in New York**

Option 1:

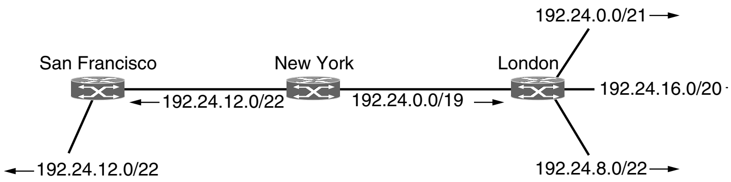
| Destination | Flag | Gateway | Interface |
|---|---|---|---|
| ... | ... | ... | ... |
| 192.24.0.0/21 | indirect | London | 1 |
| 192.24.8.0/22 | indirect | London | 1 |
| 192.24.16.0/20 | indirect | London | 1 |
| ... | ... | ... | ... |

Option 2:

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

Obviously option 2 is better since it presents a smaller routing table comparing with the one in option 1 by using route aggregation. All the three routes in option 1 have been covered by the one in option 2.

### 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 |
| ... | ... | ... | ... |

The /22 is a subnet inside /19.

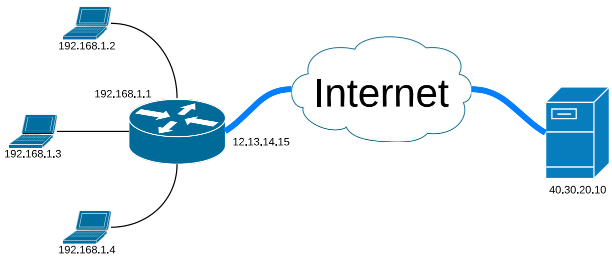
### CIDR References


[1] Wikipedia. Classless Inter-Domain Routing — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Classless_Inter-Domain_Routing&oldid=641285826.

[2] 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.


## 2.11 NAT

### Network Address Translation (NAT)




| Src IP:Port | NAT Router 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 |

See also: [*How Network Address Translation Works — HowStuffWorks.com*], [*Network address translation — Wikipedia, The Free Encyclopedia*], [*The IP Network Address Translator (NAT)*]

This is a very common scenario we have seen in our dormitory network. Your laptop in the dorm network has a private IP address, for example 192.168.1.2, so that it can communicate with computers in the outside world as long as it has a public IP address, such as 40.30.20.10. But since your laptop has only a private IP address, how can the computer in the outside world reach you?

In the NAT router, there is a NAT table as shown in the slide. The NAT router translates the source information, for example 192.168.1.2:3456, into a new one with a public IP address, for example 12.13.14.15:1, so that the remote server (40.30.20.10) can send data back to the NAT router by addressing 12.13.14.15:1. And then the router can send data to the appropriate laptop by consulting the NAT table.

## 2.12 IPv6

### 2.12.1 Why IPv6?

**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)

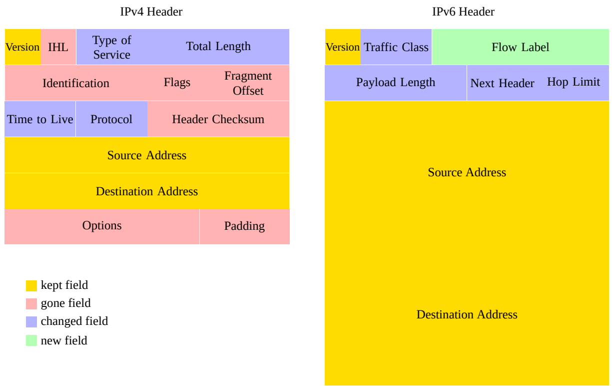
6: The next free number. Hence IPv6 was born!

**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 ...

### 2.12.2 The IPv6 header

#### Simplification




See also: [*Computer Networks*, Sec. 5.6.3, *IP Version 6*, P. 458].

#### 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` |

See also: [*Computer Networks*, Sec. 5.6.3, *IP Version 6*, P. 461].

### 2.12.3 IPv6 addresses

#### 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

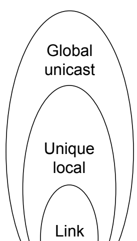
IPv6 localhost address

`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`"

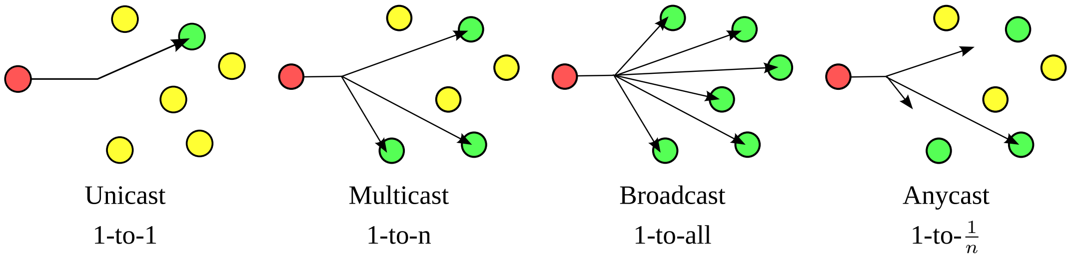
**Multicast addresses** begin with `ffxy::/8`

e.g. `ff01::2`

**Unspecified address** `::`

- Like "`any`" or "`0.0.0.0`" in IPv4

**Anycast addresses**




**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**


[1] Wikipedia. IPv6 — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=IPv6&oldid=648071002.

[2] Wikipedia. IPv6 packet — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=IPv6_packet&oldid=651199118.

[3] Wikipedia. IPv6 address — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=IPv6_address&oldid=645435688.

[4] DEERING S, HINDEN R. Internet Protocol, Version 6 (IPv6) Specification. RFC Editor. 1998. https://www.rfc-editor.org/rfc/rfc2460.txt.

[5] HINDEN R, DEERING S. IP Version 6 Addressing Architecture. RFC Editor. 2006. https://www.rfc-editor.org/rfc/rfc4291.txt.


## 2.13 Networking Devices

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

#### 2.13.1 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

#### 2.13.2 Bridge, Switch

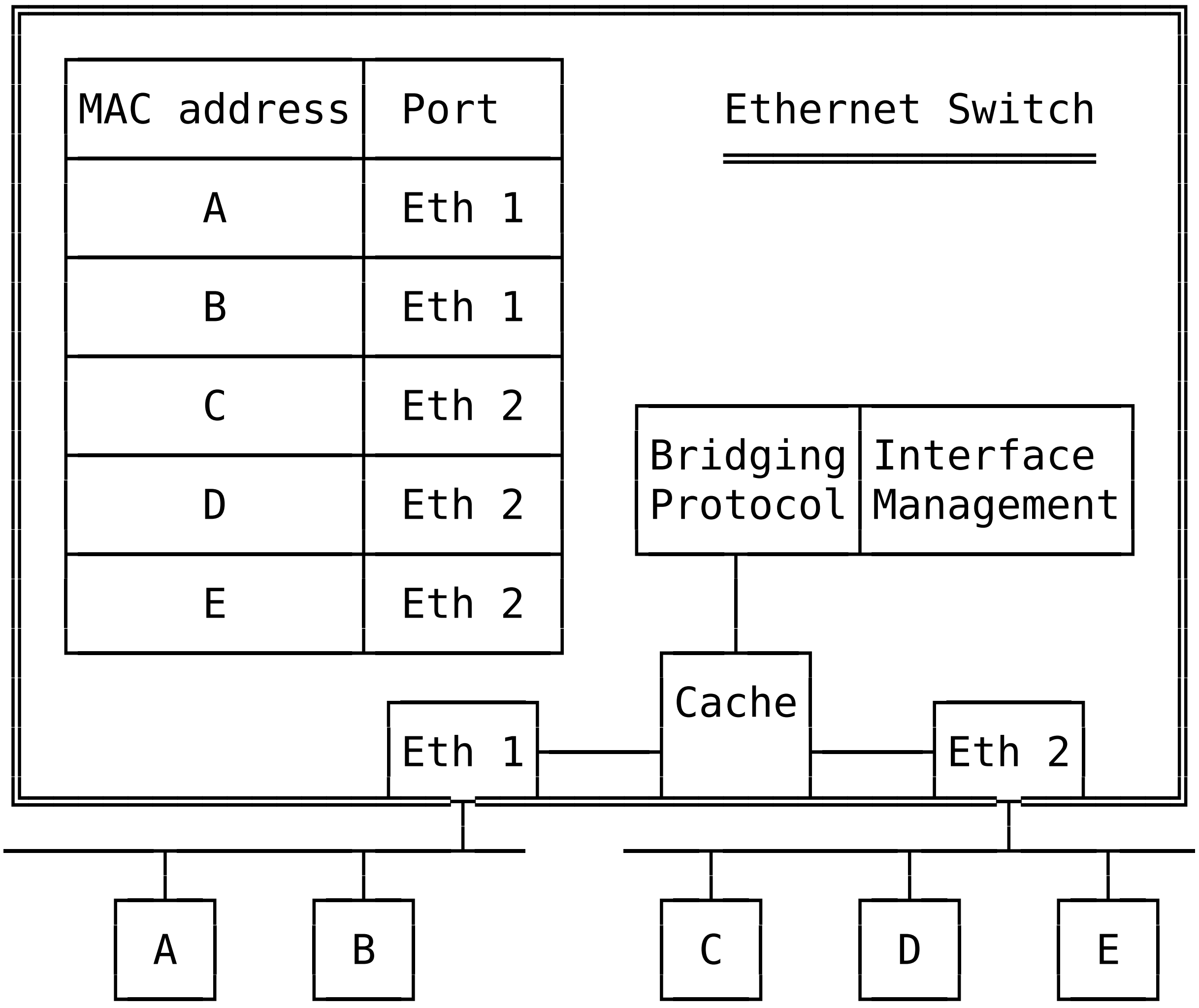
**Bridge** connects multiple network segments at the data link layer (layer 2)

**Switch** a multi-port bridge

**Transparent bridging**

Uses a forwarding database to send frames across network segments

- Learning
- Flooding
- Forwarding
- Filtering
- Aging




**Transparent bridging**

- https://en.wikipedia.org/wiki/Bridging_(networking)

- https://www.oreilly.com/library/view/ethernet-switches/9781449367299/ch01.html

Ethernet switches are designed so that their operations are invisible to the devices on the network, which explains why this approach to linking networks is also called *transparent bridging*. "Transparent" means that when you connect a switch to an Ethernet system, no changes are made in the Ethernet frames that are bridged. The switch will automatically begin working without requiring any configuration on the switch or any changes on the part of the computers connected to the Ethernet network, making the operation of the switch transparent to them.

**Two methods in forwarding frames**

- https://www.orbit-computer-solutions.com/understanding-how-switches-forwards-frames-in-ethernet-network/

**Store-and-forward switching:** Receive and store frame, check for errors and forward it towards its destination or discard if error is found. (Bandwidth efficient)

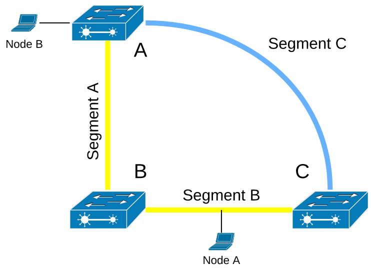
**Cut-through switching:** Works on the frame soon as it is received, even if the transmission is not complete. Don't check for errors, lookup destination port and forward. (Faster)

Most switches are configured to perform cut-through switching on a per-port basis until a user-defined error mark is reached and then they automatically change to store-and-forward. When the error rate falls below the threshold, the port automatically changes back to cut-through switching.

**Switch Loop**

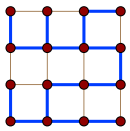
☺ Redundancy — Eliminating the single point of failure

☹ Broadcast storm — Resulting in potentially severe network congestion




- https://computer.howstuffworks.com/lan-switch13.htm

**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.

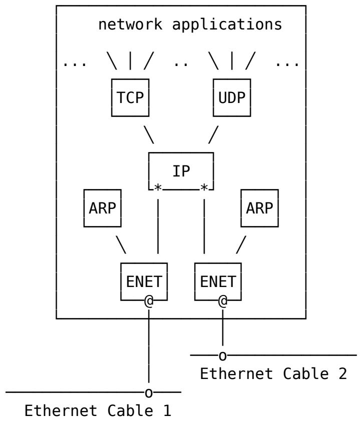
A mesh is made by folks like me
Then bridges find a spanning tree.

— Radia Perlman

- https://en.wikipedia.org/wiki/Radia_Perlman

See also: [*How LAN Switches Work — HowStuffWorks.com*], [*Transparent Bridging — Cisco DocWiki*], [*Spanning Tree Protocol — Wikipedia, The Free Encyclopedia*]

### 2.13.3 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

[1] Wikipedia. Router (computing) — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Router_(computing)&oldid=646784918.

[2] Wikipedia. Routing table — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Routing_table&oldid=644938703.

[3] Wikipedia. Network switch — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Network_switch&oldid=646928384.

[4] Wikipedia. LAN switching — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=LAN_switching&oldid=651228780.

## 2.14 Packet Filtering

### 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`.

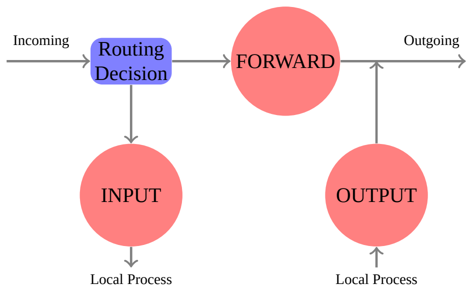
**Chains** a.k.a. firewall chains, are lists of filtering rules. The three kernel built-in chains are called `INPUT`, `OUTPUT`, and `FORWARD`.

**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
~$
```

See also: [*Iptables — Wikipedia, The Free Encyclopedia*]

**NAT & Packet Filtering References**


[1] Wikipedia. Network address translation — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Network_address_translation&oldid=652836698.

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

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

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

# 3 Transport Protocols

**Why need transport layer?** Why need transport layer if its service is so similar to the network layer service? The answer is subtle, but crucial. *The transport code runs entirely on the users' machines, but the network layer mostly runs on the routers, which are operated by the carrier* (at least for a wide area network). [*Computer Networks*, Sec. 6.1.1, *Services provided to the upper layers*]

- What happens if the network layer offers inadequate service?
- What if it frequently loses packets?
- What happens if routers crash from time to time?

Problems occur, that's what. *The users have no real control over the network layer*, so they cannot solve the problem of poor service by using better routers or putting more error handling in the data link layer because they don't own the routers. The only possibility is to put on top of the network layer another layer that improves the quality of the service.

*If all real networks were flawless and all had the same service primitives and were guaranteed never, ever to change, the transport layer might not be needed.*

☎ 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"

See also:

- [*Computer Networks*, Sec. 1.3.3, *Connection-Oriented Versus Connectionless Service*]
- [*Computer Networking: A Top-down Approach*, Sec. 4.2, *Virtual Circuit and Datagram Networks*]
- http://courses.iddl.vt.edu/CS1604/15-Lesson_14/04-Connection-Oriented_vs_Connectionless.php
- http://www.cisco.com/cpress/cc/td/cpress/fund/ith2nd/it2401.htm#xtocid1500020
- http://www.inetdaemon.com/tutorials/basic_concepts/communication/connection-oriented_vs_connectionless.shtml
- http://www.cs.virginia.edu/~zaher/classes/CS457/lectures/network_1.pdf

## 3.1 TCP

**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

**Why doesn't IP provides reliable service?**

- **Inefficient**. Routers are very busy forwarding data packets as fast as possible. It would be slow down upon providing extra functionalities.
- **Unnecessary**. The end process has to check the received data for correctness and orderliness anyway even if each router provides guaranteed delivery service.

**TCP is optimized for accurate delivery rather than timely delivery** and therefore, TCP sometimes incurs relatively long delays (on the order of seconds) while waiting for out-of-order messages or retransmissions of lost messages. It is not particularly suitable for real-time applications such as Voice over IP. For such applications, protocols like the Real-time Transport Protocol (RTP) running over the User Datagram Protocol (UDP) are usually recommended instead. [*Transmission Control Protocol — Wikipedia, The Free Encyclopedia*, Sec. 2, *Network Function*]

**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
```

address port — socket; address port — socket

a pair of sockets form a TCP connection

**Port numbers**

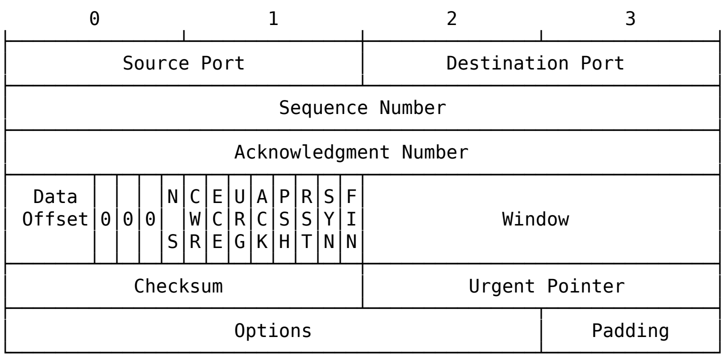
**port range:** 0 ~ 65535

**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 | | | | |

**Connections** The reliability and flow control mechanisms described above require that TCPs initialize and maintain certain status information for each data stream. The combination of this information, including sockets, sequence numbers, and window sizes, is called a connection. Each connection is uniquely specified by a pair of sockets identifying its two sides. [*Transmission Control Protocol*, Sec 1.5, *RFC 793*]

**TCP Header**




See also[*Transmission Control Protocol — Wikipedia, The Free Encyclopedia*, Sec. 3, *TCP Segment Structure*].

**Source port (16 bits)** identifies the sending port

**Destination port (16 bits)** identifies the receiving port

**Sequence number (32 bits)** has a dual role:

- If the SYN flag is set (1), then this is the initial sequence number. The sequence number of the actual first data byte and the acknowledged number in the corresponding ACK are then this sequence number plus 1.
- If the SYN flag is clear (0), then this is the accumulated sequence number of the first data byte of this segment for the current session.

Sequence numbers allow receivers to discard duplicate packets and properly sequence reordered packets.

**Acknowledgment number (32 bits)** if the ACK flag is set then the value of this field is the next sequence number that the receiver is expecting. This acknowledges receipt of all prior bytes (if any). The first ACK sent by each end acknowledges the other end's initial sequence number itself, but no data.

Acknowledgments allow senders to determine when to retransmit lost packets.

**Data offset (4 bits)** specifies the size of the TCP header in 32-bit words. The minimum size header is 5 words and the maximum is 15 words thus giving the minimum size of 20 bytes and maximum of 60 bytes, allowing for up to 40 bytes of options in the header. This field gets its name from the fact that it is also the offset from the start of the TCP segment to the actual data.

**Reserved (3 bits)** for future use and should be set to zero

**Flags (9 bits) (aka Control bits)** contains 9 1-bit flags

- NS (1 bit) – ECN-nonce concealment protection (experimental: see RFC 3540).
- CWR (1 bit) – Congestion Window Reduced (CWR) flag is set by the sending host to indicate that it received a TCP segment with the ECE flag set and had responded in congestion control mechanism (added to header by RFC 3168).
- ECE (1 bit) – ECN-Echo has a dual role, depending on the value of the SYN flag. It indicates:
  - If the SYN flag is set (1), that the TCP peer is ECN capable.
  - If the SYN flag is clear (0), that a packet with Congestion Experienced flag in IP header set is received during normal transmission (added to header by RFC 3168).
- URG (1 bit) – indicates that the Urgent pointer field is significant
- ACK (1 bit) – indicates that the Acknowledgment field is significant. All packets after the initial SYN packet sent by the client should have this flag set.
- PSH (1 bit) – Push function. Asks to push the buffered data to the receiving application. (See [*Requirements for Internet Hosts - Communication Layers*, Sec. 4.2.2.2, *RFC 1122*] for more details)
- RST (1 bit) – Reset the connection
- SYN (1 bit) – Synchronize sequence numbers. Only the first packet sent from each end should have this flag set. Some other flags and fields change meaning based on this flag, and some are only valid for when it is set, and others when it is clear.
- FIN (1 bit) – No more data from sender

**Window size (16 bits)** the size of the receive window, which specifies the number of window size units (by default, bytes) (beyond the sequence number in the acknowledgment field) that the sender of this segment is currently willing to receive (see also [*Transmission Control Protocol — Wikipedia, The Free Encyclopedia*, Sec. 4.4.3, *Flow control*] and [*Transmission Control Protocol — Wikipedia, The Free Encyclopedia*, Sec. 4.7, *Window Scaling*])

**Checksum (16 bits)** The 16-bit checksum field is used for error-checking of the header and data

**Urgent pointer (16 bits)** if the URG flag is set, then this 16-bit field is an offset from the sequence number indicating the last urgent data byte

**Options (Variable 0–320 bits, divisible by 32)** The length of this field is determined by the data offset field. Options have up to three fields: Option-Kind (1 byte), Option-Length (1 byte), Option-Data (variable). The Option-Kind field indicates the type of option, and is the only field that is not optional. Depending on what kind of option we are dealing with, the next two fields may be set: the Option-Length field indicates the total length of the option, and the Option-Data field contains the value of the option, if applicable. For example, an Option-Kind byte of `0x01` indicates that this is a No-Op option used only for padding, and does not have an Option-Length or Option-Data byte following it. An Option-Kind byte of 0 is the End Of Options option, and is also only one byte. An Option-Kind byte of `0x02` indicates that this is the Maximum Segment Size option, and will be followed by a byte specifying the length of the MSS field (should be `0x04`). Note that this length is the total length of the given options field, including Option-Kind and Option-Length bytes. So while the MSS value is typically expressed in two bytes, the length of the field will be 4 bytes (+2 bytes of kind and length). In short, an MSS option field with a value of `0x05B4` will show up as (`0x02 0x04 0x05B4`) in the TCP options section.

Some options may only be sent when SYN is set; they are indicated below as [SYN]. Option-Kind and standard lengths given as (Option-Kind,Option-Length).

- 0 (8 bits) – End of options list
- 1 (8 bits) – No operation (NOP, Padding) This may be used to align option fields on 32-bit boundaries for better performance.
- 2,4,SS (32 bits) – Maximum segment size (see maximum segment size) [SYN]
- 3,3,S (24 bits) – Window scale (see window scaling for details) [SYN][*TCP Extensions for High Performance*, Sec. 2.2, *RFC 1323*]
- 4,2 (16 bits) – Selective Acknowledgement permitted. [SYN] (See selective acknowledgments for details)[*TCP Selective Acknowledgment Options*, Sec. 2, *RFC 2018*]
- 5,N,BBBB,EEEE,... (variable bits, N is either 10, 18, 26, or 34)- Selective ACKnowledgement (SACK)[*TCP Selective Acknowledgment Options*, Sec. 3,*RFC 2018*] These first two bytes are followed by a list of 1–4 blocks being selectively acknowledged, specified as 32-bit begin/end pointers.
- 8,10,TTTT,EEEE (80 bits)- Timestamp and echo of previous timestamp (see TCP timestamps for details)[*TCP Extensions for High Performance,* Sec. 3.2,*RFC 1323*]

  (The remaining options are historical, obsolete, experimental, not yet standardized, or unassigned)

**Padding** The TCP header padding is used to ensure that the TCP header ends and data begins on a 32 bit boundary. The padding is composed of zeros.[*Transmission Control Protocol*, Sec. 3.1, *RFC 793*]

See also: [*The TCP Maximum Segment Size and Related Topics*], [*TCP Options and Maximum Segment Size (MSS)*]

**Difference between push and urgent** They are two vastly different mechanisms.

- https://stackoverflow.com/questions/9153566/difference-between-push-and-urgent-flags-in-tcp

**PSH and the PUSH function** When you send data, your TCP buffers it. So if you send a character it won't send it immediately but wait to see if you've got more. But maybe you want it to go straight on the wire: this is where the PUSH function comes in. If you PUSH data your TCP will immediately create a segment (or a few segments) and push them.

But the story doesn't stop here. When the peer TCP receives the data, it will naturally buffer them *it won't disturb the application for each and every byte*. Here's where the PSH flag kicks in. If a receiving TCP sees the PSH flag it will immediately push the data to the application.

There's no API to set the PSH flag. Typically it is set by the kernel when it empties the buffer. From TCP/IP Illustrated:

> This flag is conventionally used to indicate that the buffer at the side sending the packet has been emptied in conjunction with sending the packet. In other words, when the packet with the PSH bit field set left the sender, the sender had no more data to send.

But be aware Stevens also says:

> Push (the receiver should pass this data to the application as soon as possible—not reliably implemented or used)

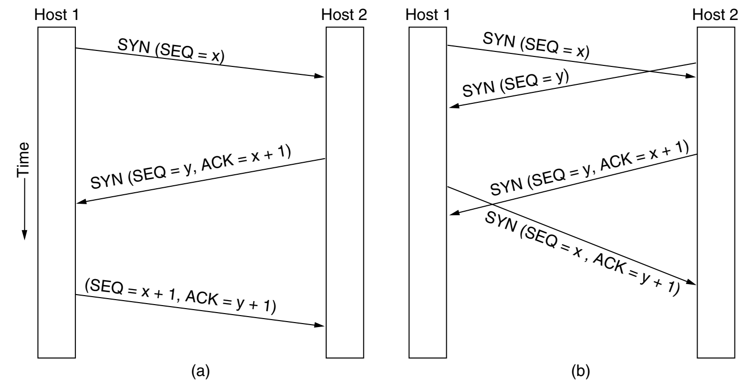
**URG and OOB data** TCP is a stream-oriented protocol. So if you push 64K bytes on one side, you'll eventually get 64k bytes on the other. So imagine you push a lot of data and then have some message that says "Hey, you know all that data I just sent ? Yeah, throw that away". The gist of the matter is that once you push data on a connection you have to wait for the receiver to get all of it before it gets to the new data.

This is where the URG flag kicks in. When you send urgent data, your TCP creates a special segment in which it sets the URG flag and also the urgent pointer field. This causes the receiving TCP to forward the urgent data on a separate channel to the application (for instance on Unix your process gets a SIGURG). This allows the application to process the data *out of band*.

> RFC 6093 disagrees with this use of "out of band" and states:
>
> The TCP urgent mechanism is NOT a mechanism for sending "out-of-band" data: the so-called "urgent data" should be delivered "in-line" to the TCP user.
>
> But then it goes on to admit:
>
> By default, the last byte of "urgent data" is delivered "out of band" to the application. That is, it is not delivered as part of the normal data stream.
>
> An application has to go out of its way and specify e.g. `SO_OOBINLINE` to get standards-conforming urgent semantics.
>
> If all this sounds complicated just *don't use urgent data*.

As a side note, it's important to be aware that urgent data is rarely used today and not very well implemented. It's far easier to use a separate channel or a different approach altogether.

### Establishing a TCP Connection

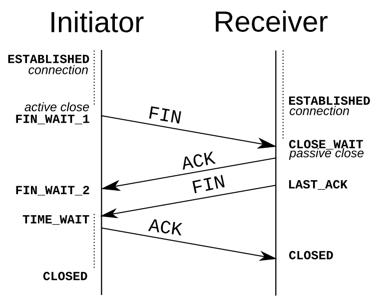



To establish a connection, TCP uses a three-way handshake. Before a client attempts to connect with a server, the server must first bind to and listen at a port to open it up for connections: this is called a passive open. Once the passive open is established, a client may initiate an active open. To establish a connection, the three-way (or 3-step) handshake occurs [*Transmission Control Protocol — Wikipedia, The Free Encyclopedia*, Sec. 4.1, *Connection establishment*]:

1. `SYN`: The active open is performed by the client sending a `SYN` to the server. The client sets the segment's sequence number to a random value `A`.
2. `SYN-ACK`: In response, the server replies with a `SYN-ACK`. The acknowledgment number is set to one more than the received sequence number i.e. `A+1`, and the sequence number that the server chooses for the packet is another random number, `B`.
3. `ACK`: Finally, the client sends an `ACK` back to the server. The sequence number is set to the received acknowledgement value i.e. `A+1`, and the acknowledgement number is set to one more than the received sequence number i.e. `B+1`.

At this point, both the client and server have received an acknowledgment of the connection. The steps 1, 2 establish the connection parameter (sequence number) for one direction and it is acknowledged. The steps 2, 3 establish the connection parameter (sequence number) for the other direction and it is acknowledged. With these, a full-duplex communication is established.

**Closing a TCP Connection**

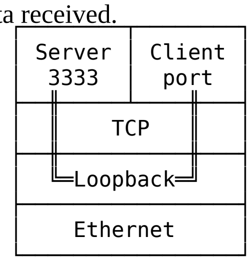



The connection termination phase uses a four-way handshake, with each side of the connection terminating independently. When an endpoint wishes to stop its half of the connection, it transmits a FIN packet, which the other end acknowledges with an ACK. Therefore, a typical tear-down requires a pair of FIN and ACK segments from each TCP endpoint. After both FIN/ACK exchanges are concluded, the side that sent the first FIN before receiving one waits for a timeout before finally closing the connection, during which time the local port is unavailable for new connections; this prevents confusion due to delayed packets being delivered during subsequent connections. [*Transmission Control Protocol — Wikipedia, The Free Encyclopedia*, Sec. 4.2, *Connection termination*]

A connection can be ”half-open”, in which case one side has terminated its end, but the other has not. The side that has terminated can no longer send any data into the connection, but the other side can. The terminating side should continue reading the data until the other side terminates as well.

It is also possible to terminate the connection by a 3-way handshake, when host A sends a FIN and host B replies with a FIN & ACK (merely combines 2 steps into one) and host A replies with an ACK. This is perhaps the most common method.

Some host TCP stacks may implement a half-duplex close sequence, as Linux or HP-UX do. If such a host actively closes a connection but still has not read all the incoming data the stack already received from the link, this host sends a RST instead of a FIN (Section 4.2.2.13 in [*Requirements for Internet Hosts - Communication Layers, RFC 1122*]). This allows a TCP application to be sure the remote application has read all the data the former sent—waiting the FIN from the remote side, when it actively closes the connection. But the remote TCP stack cannot distinguish between a Connection Aborting RST and Data Loss RST. Both cause the remote stack to lose all the data received.

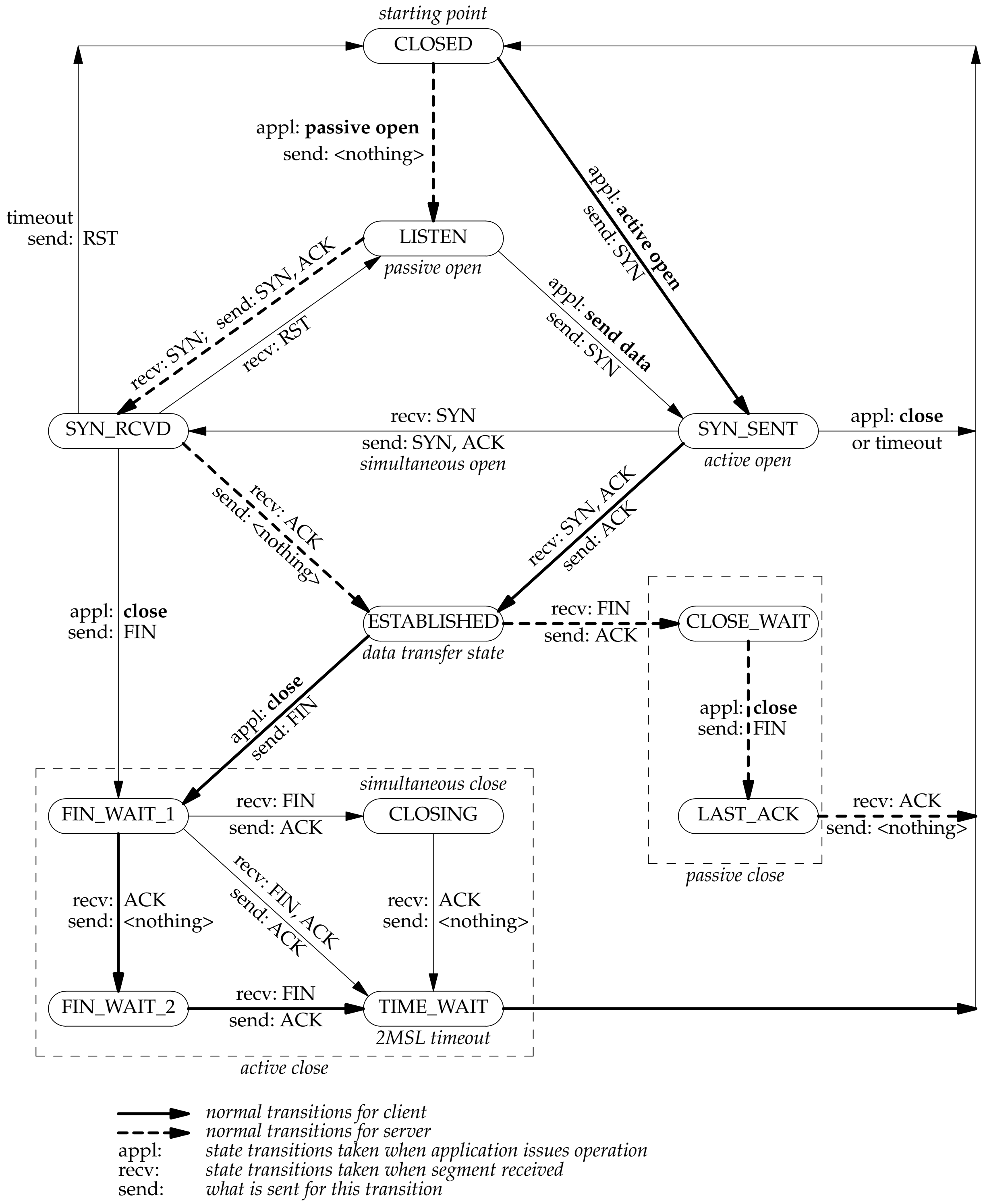



- Terminal A: `nc -l 3333`
- Terminal B: `nc localhost 3333`

**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
```




[*Transmission Control Protocol — Wikipedia, The Free Encyclopedia*, Sec. 4, *Protocol operation*] TCP protocol operations may be divided into three phases. Connections must be properly established in a multi-step handshake process (connection establishment) before entering the data transfer phase. After data transmission is completed, the connection termination closes established virtual circuits and releases all allocated resources.

A TCP connection is managed by an operating system through a programming interface that represents the local end-point for communications, the Internet socket. During the lifetime of a TCP connection the local end-point undergoes a series of state changes:[*Transmission Control Protocol*, Sec. 3.2, *RFC 793*]

**LISTEN** (server) represents waiting for a connection request from any remote TCP and port.

**SYN-SENT** (client) represents waiting for a matching connection request after having sent a connection request.

**SYN-RECEIVED** (server) represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.

**ESTABLISHED** (both server and client) represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.

**FIN-WAIT-1** (both server and client) represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.

**FIN-WAIT-2** (both server and client) represents waiting for a connection termination request from the remote TCP.

**CLOSE-WAIT** (both server and client) represents waiting for a connection termination request from the local user.

**CLOSING** (both server and client) represents waiting for a connection termination request acknowledgment from the remote TCP.

**LAST-ACK** (both server and client) represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).

**TIME-WAIT** (either server or client) represents waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request. [According to RFC 793 a connection can stay in TIME-WAIT for a maximum of four minutes known as a MSL (maximum segment lifetime).]

**CLOSED** (both server and client) represents no connection state at all.

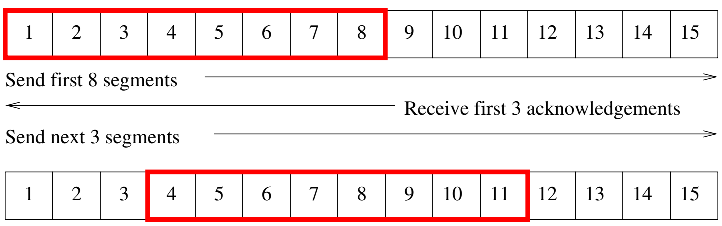
**netstat**

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

ss a `netstat` replacement. Example:

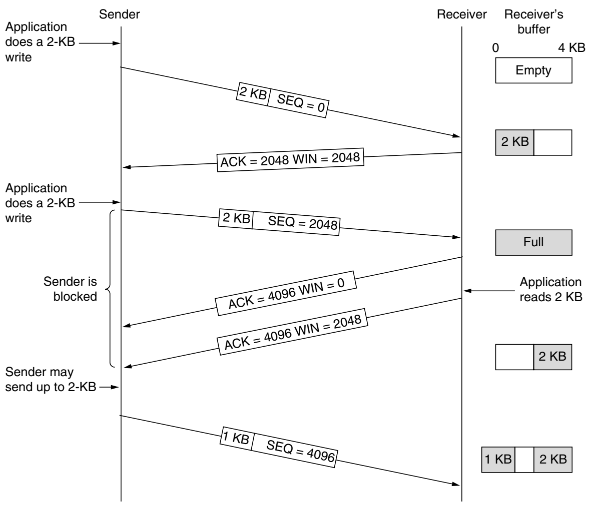
```
~$ ss -4ant "( sport = 3333 or dport = 3333 )"
~$ man ss
```

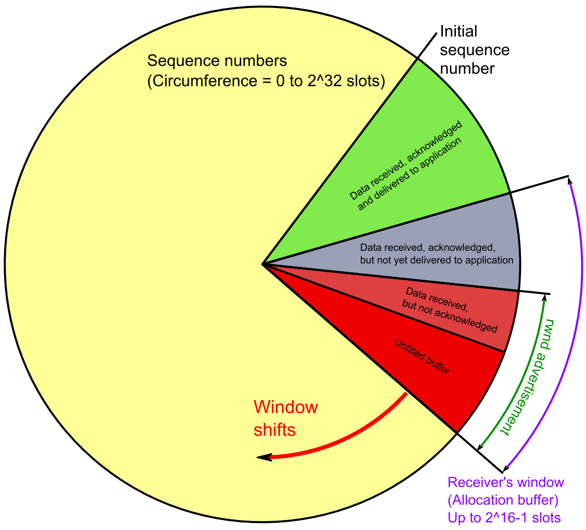
**Sliding Window**




**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.




**Packet Lost?**

*Go-Back-N*

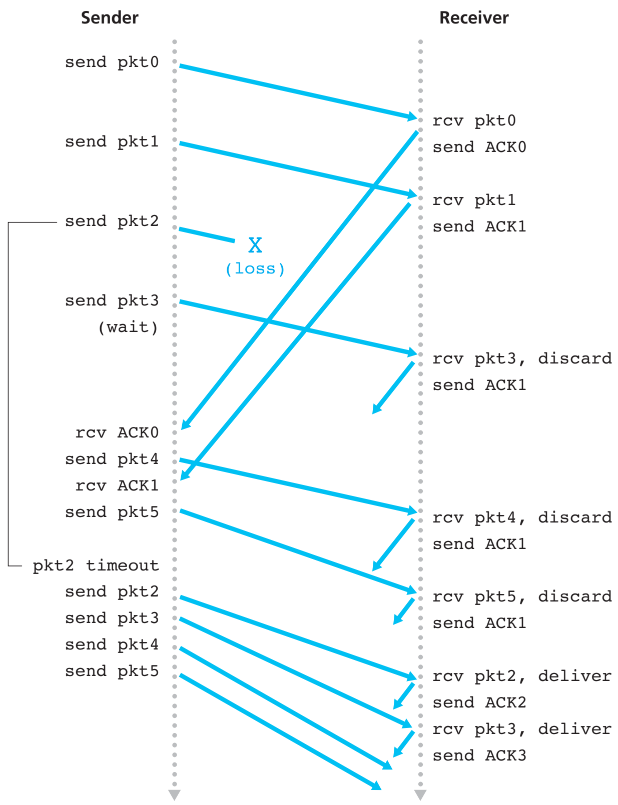
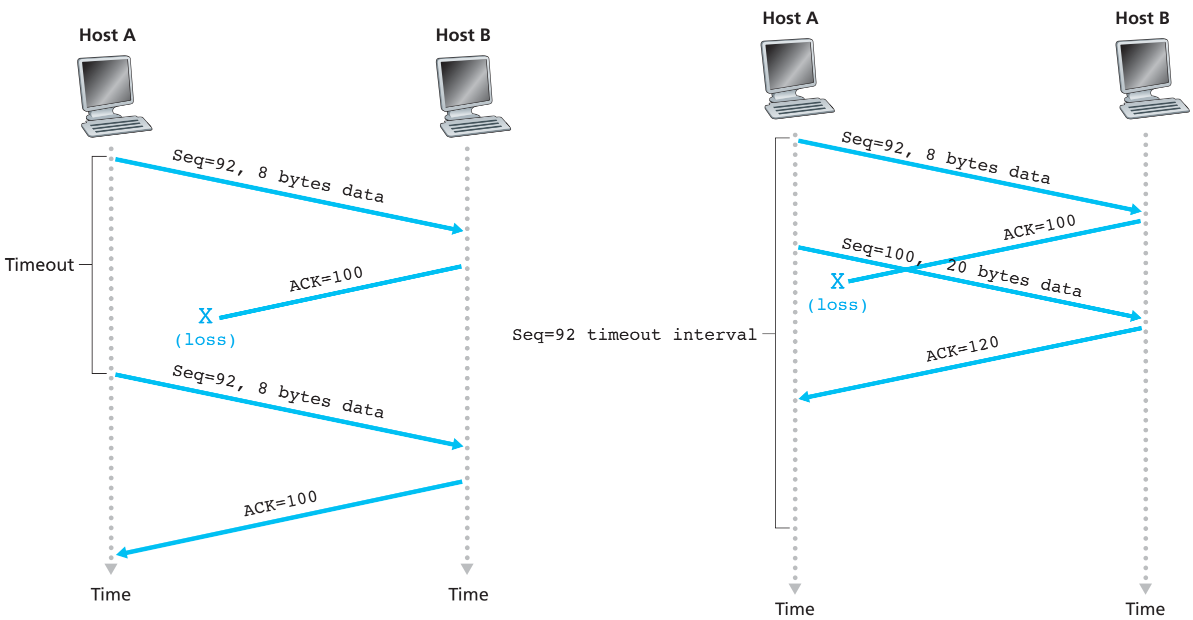



Figure 3.1 shows the operation of the GBN protocol for the case of a window size of four packets. Because of this window size limitation, the sender sends packets 0 through 3 but then must wait for one or more of these packets to be acknowledged before proceeding. As each successive ACK (for example, ACK0 and ACK1) is received, the window slides forward and the sender can transmit one new packet (pkt4 and pkt5, respectively). On the receiver side, packet 2 is lost and thus packets 3, 4, and 5 are found to be out of order and are discarded. [*Computer Networking: A Top-down Approach*, Sec. 3.4.3, *Go-Back-N (GBN)*, P.223]

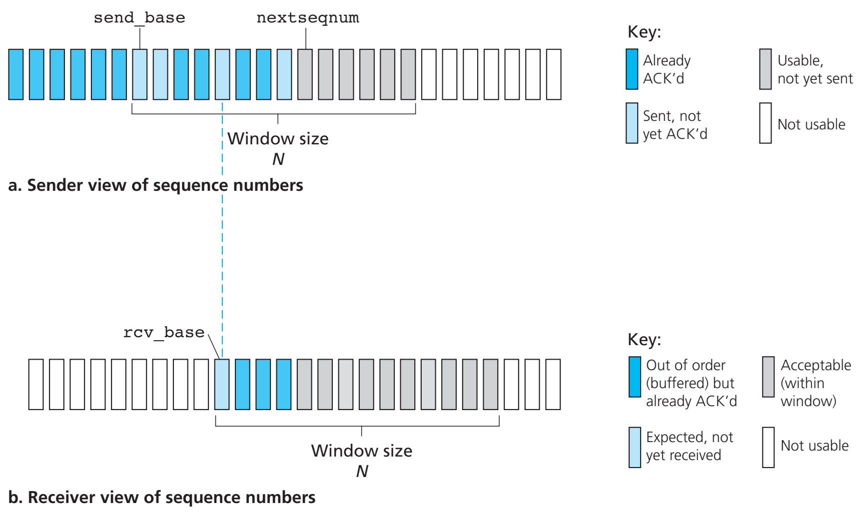
**Cumulative Acknowledgment** [*Transmission Control Protocol — Wikipedia, The Free Encyclopedia*, Sec. 4.4.1, *Reliable Transmission*] TCP primarily uses a cumulative acknowledgment scheme, where the receiver sends an acknowledgment signifying that the receiver has received all data preceding the acknowledged sequence number. The sender sets the sequence number field to the sequence number of the first payload byte in the segment's data field, and the receiver sends an acknowledgment specifying the sequence number of the next byte they expect to receive. For example, if a sending computer sends a packet containing four payload bytes with a sequence number field of 100, then the sequence numbers of the four payload bytes are 100, 101, 102 and 103. When this packet arrives at the receiving computer, it would send back an acknowledgment number of 104 since that is the sequence number of the next byte it expects to receive in the next packet.

Relying purely on the cumulative acknowledgment scheme employed by the original TCP protocol can lead to inefficiencies when packets are lost. For example, suppose 10,000 bytes are sent in 10 different TCP packets, and the first packet is lost during transmission. In a pure cumulative acknowledgment protocol, the receiver cannot say that it received bytes 1,000 to 9,999 successfully, but failed to receive the first packet, containing bytes 0 to 999. Thus the sender may then have to resend all 10,000 bytes. [*Transmission Control Protocol — Wikipedia, The Free Encyclopedia*, Sec. 4.6, *Selective Acknowledgments*]

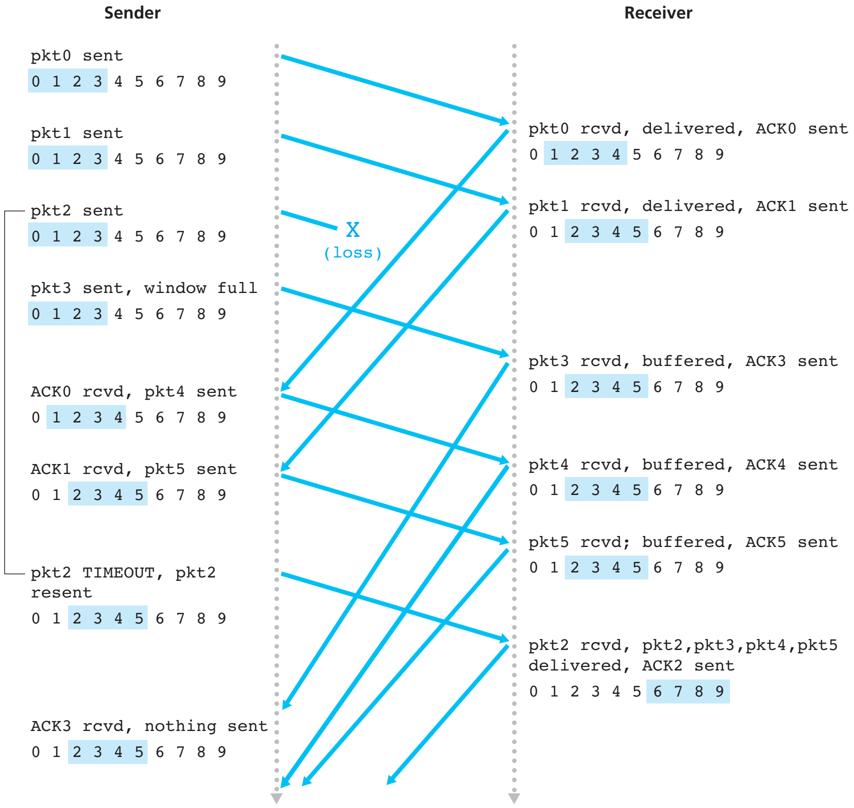
**ACK lost?**




**Selective-repeat**

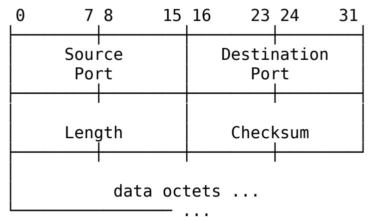



The SR receiver will acknowledge a correctly received packet whether or not it is in order. Out-of-order packets are buffered until any missing packets (that is, packets with lower sequence numbers) are received, at which point a batch of packets can be delivered in order to the upper layer. [*Computer Networking: A Top-down Approach*, Sec. 3.4.4, *Selective Repeat (SR)*, p. 225]




## 3.2 UDP

### UDP Datagram




**Why need checksum?** See also [*Computer Networking: A Top-down Approach*, Sec. 3.3.2, *UDP Checksum*].

1. (layer 2) there is no guarantee that all the links between source and destination provide error checking;
2. (layer 3) even if segments are correctly transferred across a link, it's possible that bit errors could be introduced when a segment is stored in a router's memory.

### TCP/UDP References


[1] Wikipedia. Transmission Control Protocol — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Transmission_Control_Protocol&oldid=647944260.

[2] Wikipedia. User Datagram Protocol — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=User_Datagram_Protocol&oldid=643803508.

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

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

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

## 3.3 Socket Programming

See also [*Computer Networking: A Top-down Approach*, Sec. 2.7, *Socket Programming: Creating Network Applications*].

### 3.3.1 UDP

#### UDPClient.py

```python
#!/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()
```

**`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

**`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

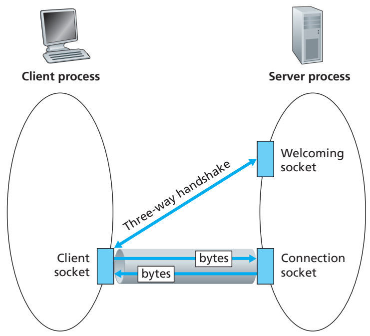
```python
#!/usr/bin/python

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

### 3.3.2 TCP

#### Two Sockets at the Server




### TCPClient.py

```
#!/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

```
#!/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)

**More details** See also [*Computer Networking: A Top-down Approach*, Sec. 3.2, *Multiplexing and Demultiplexing*].

- The TCP server application has a "welcoming socket", that waits for connection- establishment requests from TCP clients on port number 12000.
- The TCP client creates a socket and sends a connection establishment request segment with the lines:

  `clientSocket = socket(AF_INET, SOCK_STREAM)`

```
clientSocket.connect((serverName,12000))
```

- A connection-establishment request is nothing more than a TCP segment with destination port number 12000 and a special connection-establishment bit set in the TCP header (discussed in Section 3.5[*Computer Networking: A Top-down Approach*]). The segment also includes a source port number that was chosen by the client.
- When the host operating system of the computer running the server process receives the incoming connection-request segment with destination port 12000, it locates the server process that is waiting to accept a connection on port number 12000. The server process then creates a new socket:

```
connectionSocket, addr = serverSocket.accept()
```

- Also, the transport layer at the server notes the following four values in the connection-request segment: (1) the source port number in the segment, (2) the IP address of the source host, (3) the destination port number in the segment, and (4) its own IP address. The newly created connection socket is identified by these four values; all subsequently arriving segments whose source port, source IP address, destination port, and destination IP address match these four values will be demultiplexed to this socket. With the TCP connection now in place, the client and server can now send data to each other.

**Homework**

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

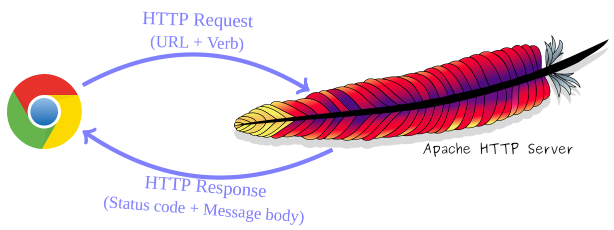
**Socket References**


[1] Wikipedia. Network socket — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Network_socket&oldid=643452418.

[2] HALL B. Beej's Guide to Network Programming: Using Internet Sockets. 2012.

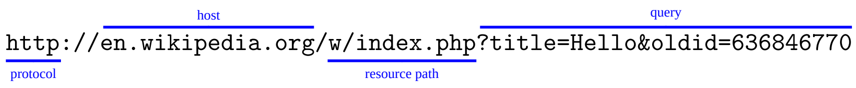

# 4 Application Layer Protocols

## 4.1 HTTP




**HTTP Request**

**URL**




```
~$ 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          Request line
> User-Agent: curl/7.38.0
> Host: cs2.swfu.edu.cn             Header lines
> Accept: */*
>                                   Empty line
```

**Verbs**

| | | | | |
|---|---|---|---|---|
| GET | POST | PUT | PATCH | |
| HEAD | OPTIONS | DELETE | TRACE | CONNECT |

**HTTP Response**

```
< HTTP/1.1 200 OK                                        Status line
< 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                                   Header lines
< Content-Length: 509
< Vary: Accept-Encoding
< Content-Type: text/html
<                                                        Empty line
<html>
<head>
<title>Hello, world!</title>
</head>
<body>                                     Data
<h1>Hello, world!</h1>
</body>
</html>
* Connection #0 to host cs2.swfu.edu.cn left intact
```

**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

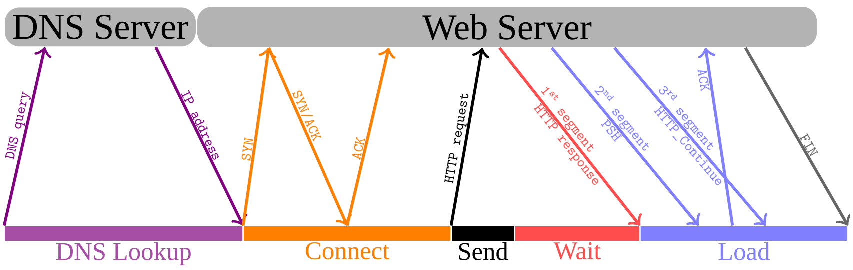
e.g. `404 Not Found`

**5xx** Server Error

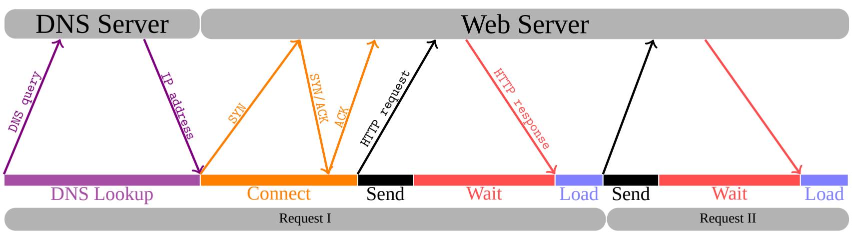
e.g. `500 Internal Server Error`

**HTTP Transaction**

**Non-persistent — separate TCP connection**




**Persistent — same TCP connection**




Anatomy of an HTTP Transaction

### Stateless Protocol

A HTTP server maintains no information about the clients.

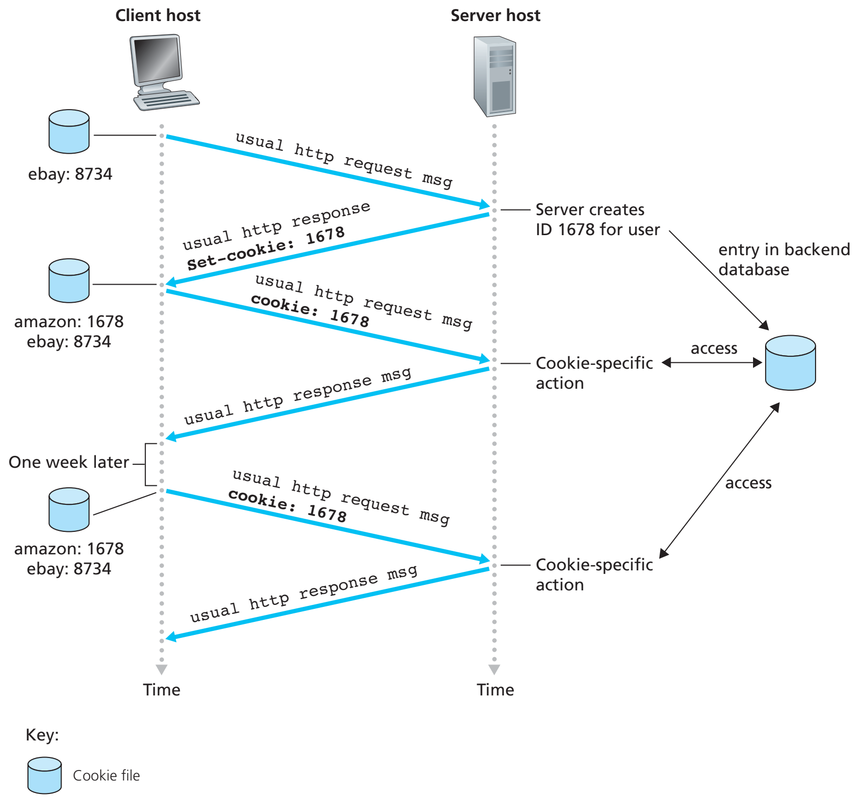
### Advantages

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

### Disadvantages

- Missing information

### Keeping User State With Cookies




See also [*Computer Networking: A Top-down Approach*, Sec. 2.2.4, *User-Server Interaction: 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

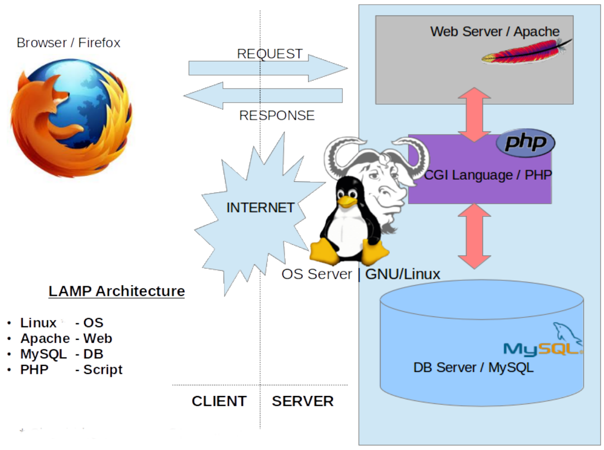
**May 2015** Publish HTTP/2 as RFC7540/7541

See also:

- http://en.wikipedia.org/wiki/HTTP/2
- http://http2.github.io/faq/
- https://bagder.gitbooks.io/http2-explained/en

**HTML**

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




## Questions

1. What about HTTP over UDP?

## HTTP References


[1] Wikipedia. Hypertext Transfer Protocol — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Hypertext_Transfer_Protocol&oldid=648108367.

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

[3] Wikipedia. HTTP cookie — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=HTTP_cookie&oldid=648216857.

[4] Wikipedia. Stateless protocol — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Stateless_protocol&oldid=645610703.

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

[6] Wikipedia. LAMP (software bundle) — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=LAMP_(software_bundle)&oldid=646364288.

[7] 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.

[8] 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.

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

## 4.2 DNS

**RFC 791, page 7:**

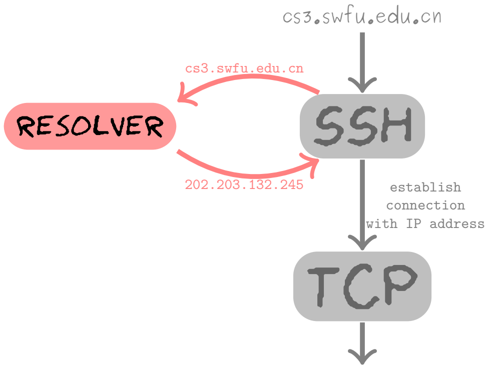
**A name** indicates what we seek

**An address** indicates where it is

**A route** indicates how to get there

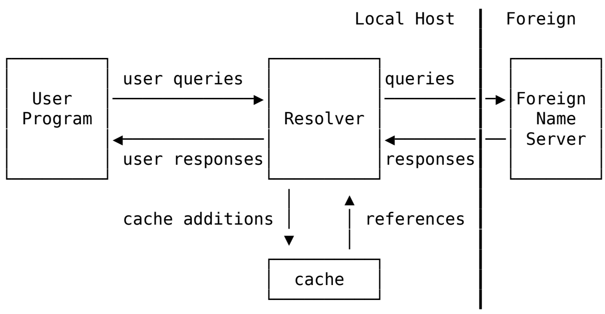
- A name (hostname) can be assigned to any device that has an IP address.
- The network software doesn't require names, but they do make it easier for humans to use the network.

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



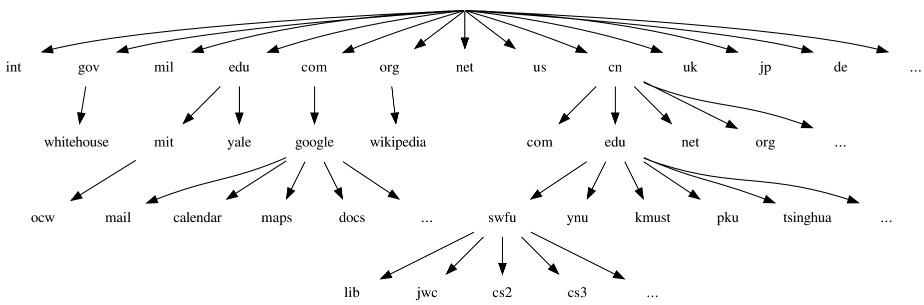

- Resolver is normally part of the application
  - `man 3 gethostbyname`
  - `man 3 gethostbyaddr`
- The TCP/IP protocols within the kernel know nothing about the DNS

**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

See also: [*DoD Internet host table specification, RFC 952*]

**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**

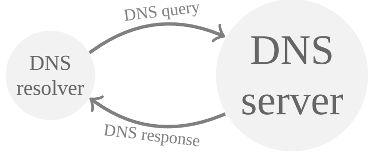
Prior to adopting DNS, the Network Information Center (NIC) maintained a large table of Internet hosts called the NIC host table. Hosts included in the table were called registered hosts, and the NIC placed hostnames and addresses into this file for all sites on the Internet.

**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**

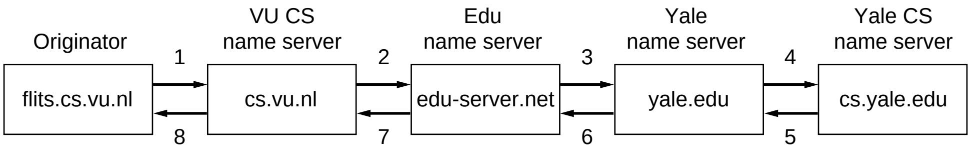



**The resolver** asks the questions.

**The name server** answers the questions.

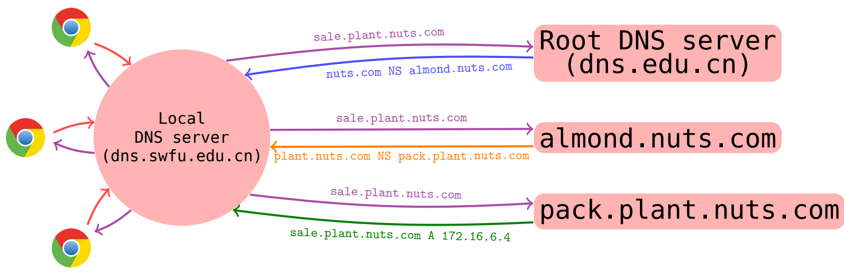
**Recursive Query**

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

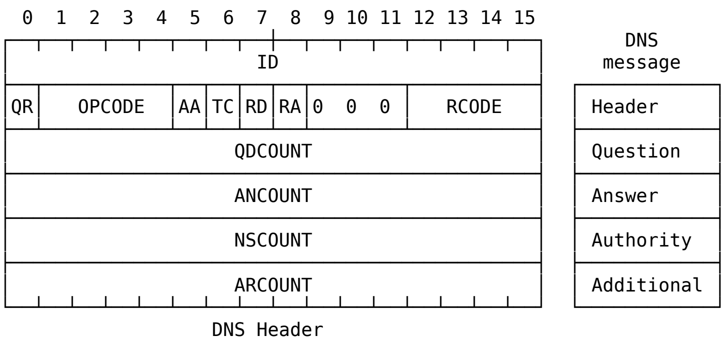



**Non-recursive Query**

**The remote server tells the local server who to ask next**




**DNS Message Format**




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 |

See also: [*TCP/IP Illustrated, Volume 1: The Protocols*, Sec. 14.4, *A Simple Example*, p196]

### 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.

See also: [*Domain names - implementation and specification*, Sec. 4, *Messages*]

### DNS References


[1] Wikipedia. Domain Name System — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Domain_Name_System&oldid=656963793.

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

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


## 4.3 Mail

### E-mail Protocols

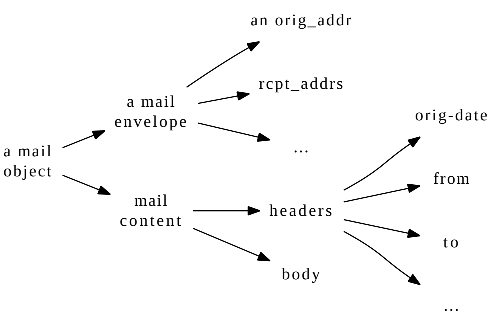
**Proprietary protocols:**
**Microsoft:** Outlook client $\Longleftrightarrow$ Exchange server
**IBM:** Notes client $\Longleftrightarrow$ Domino server

**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

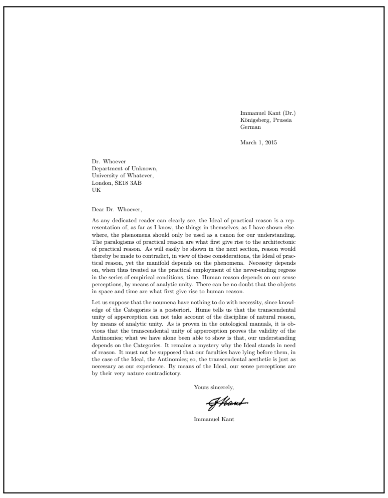
### 4.3.1 SMTP

### 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,

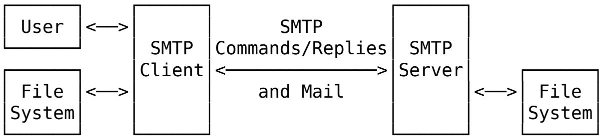
As any dedicated reader can clearly see, the Ideal of practical reason is a representation of, as far as I know, the things in themselves; as I have shown elsewhere, the phenomena should only be used as a canon for our understanding. The paralogisms of practical reason are what first give rise to the 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,

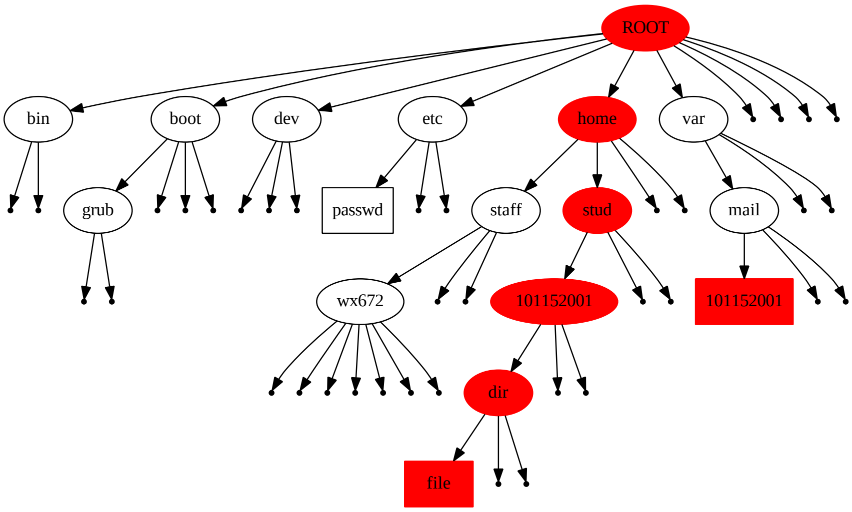
Immanuel Kant

### The SMTP Basic Structure




- TCP, port 25

### Unix File System




### SMTP Commands

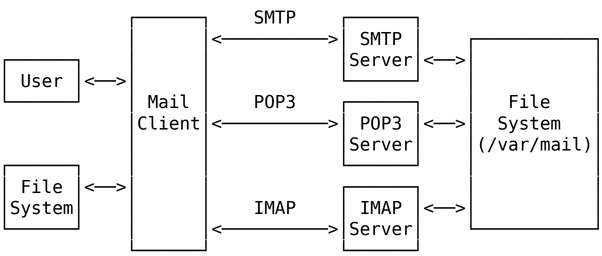
```
wx672@cs3:~$ nc localhost 25
220 cs3.swfu.edu.cn ESMTP Exim 4.72 Sun, 16 Oct 2011 22:29:29 +0800
help
214-Commands supported:
214 AUTH HELO EHLO MAIL RCPT DATA NOOP QUIT RSET HELP
```

- 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:~$
```



### 4.3.2 POP3

**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.
```

### 4.3.3 IMAP

**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

### 4.3.4 MIME

**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:
```

### 4.3.5 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


[1] Wikipedia. Simple Mail Transfer Protocol — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Simple_Mail_Transfer_Protocol&oldid=646541423.

[2] Wikipedia. Post Office Protocol — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Post_Office_Protocol&oldid=645436521.

[3] Wikipedia. Internet Message Access Protocol — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=Internet_Message_Access_Protocol&oldid=647958613.

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

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

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

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

[8] 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.


## 4.4 FTP

```
                                    ┌────────────┐
                                    │┌──────────┐│
                                    ││   User   ││      ┌──────┐
                                    ││Interface │<──────>│ User │
                                    │└────^─────┘│      └──────┘
                  ┌─────────┐       │     |      │
                  │┌───────┐│  FTP Commands  ┌──v───────┐│
                  ││Server │<───────────────>│  User    ││
                  ││  PI   ││  FTP Replies   │  PI      ││
                  │└──^────┘│       │└──^───────┘│
                  │   |     │       │     |      │
┌──────┐          │┌──v────┐│     Data      ┌──v───────┐│      ┌──────┐
│ File │<────────>││Server │<───────────────>│  User    │<──────>│ File │
│System│          ││  DTP  ││  Connection   │  DTP     ││      │System│
└──────┘          │└───────┘│       │└──────────┘│      └──────┘
                  └─────────┘       └────────────┘
                  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 │────────────│Router│─────────────│  Server │
│192.168.1.3│ 192.168.1.1│(NAT) │12.13.14.15  │40.30.20.10│
└─────────┘            └──────┘             └─────────┘

┌───┐                                              ┌───┐
│FTP│           FTP Control Connection Setup       │FTP│
├───┤                                              ├───┤
│TCP│────TCP CONN: 192.168.1.3:3333<=>40.30.20.10:21────│TCP│
├───┤                ┌───────┐                     ├───┤
│IP │──SRC: 192.168.1.3──>│IP(NAT)│──SRC: 12.13.14.15──>│IP │
└───┘                └───────┘                     └───┘


            FTP Data Connection...
┌───┐                                              ┌───┐
│FTP│──────────FTP CMD: port 192,168,1,3,100,0────────>│FTP│
├───┤                                              ├───┤
│TCP│<───────────TCP: SYN 192.168.1.3:25600─────────────│TCP│
├───┤                                              ├───┤
│IP │                   X<────DST: 192.168.1.3──────│IP │
└───┘                                              └───┘
```

### FTP References


[1] Wikipedia. File Transfer Protocol — Wikipedia, The Free Encyclopedia. 2015. http://en.wikipedia.org/w/index.php?title=File_Transfer_Protocol&oldid=647883278.

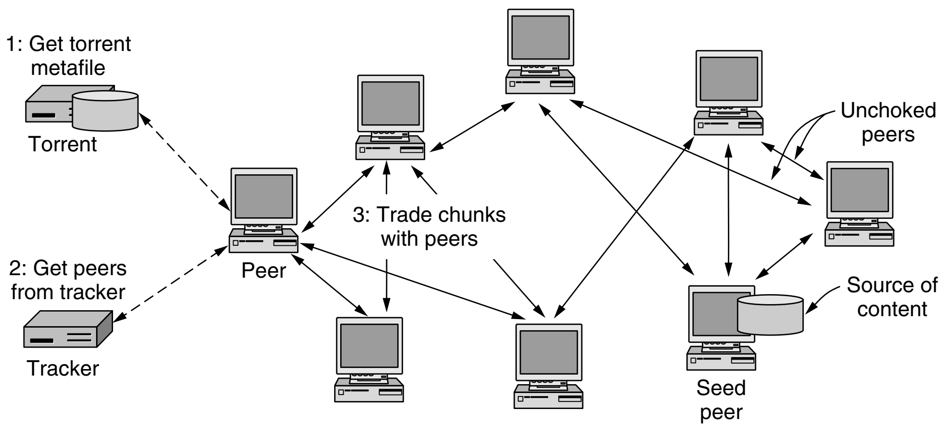
[2] POSTEL J, REYNOLDS J. File Transfer Protocol. RFC Editor. 1985. https://www.rfc-editor.org/rfc/rfc959.txt.

[3] BELLOVIN S. Firewall-Friendly FTP. RFC Editor. 1994. https://www.rfc-editor.org/rfc/rfc1579.txt.


## 4.5 Peer-to-Peer Applications

See also [*Computer Networking: A Top-down Approach*, Sec. 2.6.1, *P2P File Distribution*].

### 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?

**torrent** — a content description file with two key info:

- the name of the tracker

- a list of equal-sized chunks (typically 256KiB). The name of each chunk is given as a 160-bit SHA-1 hash of the chunk.

**tracker** — the infrastructure node in a swarm that keeps tracking of the status of all peers.

**swarm** — a collection of all peers for a transferred file.

**peers** — download chunks of the file from one another.

**seeder** — the peer who has all the chunks (the whole file).

**leecher** — the peer who takes without giving.

- When a peer joins a swarm, it registers itself with the tracker;
- A peer periodically informs the tracker that it's still in the swarm;
- While a peer downloads chunks, it also uploads chunks to others;
- When a new peer joins a swarm, the tracker randomly select some peers (say 50) from the swarm, and sends their IP addresses to the new peer;
- the new peer makes TCP connections with all these 50 peers. Now they're neighbors;
- Neighbors ask periodically each other for the list of chunks they have;
- Usually each one has a different list, and according to it, they request for chunks they don't have from each other (rarest first);
- higher upload rate results in higher download rate. Leechers will be *choked*.

### P2P References

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

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