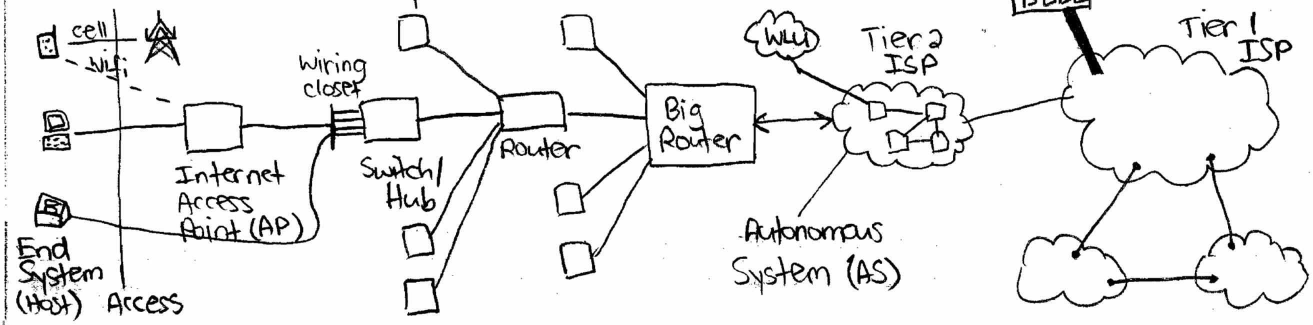
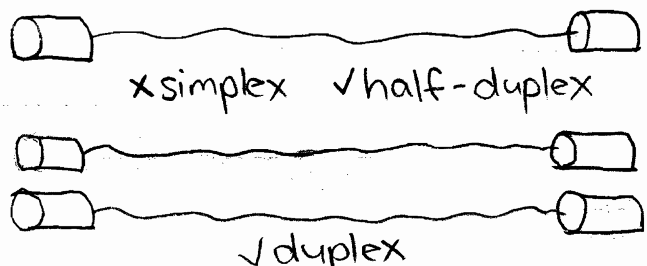
CS 436

03 Jan 2012

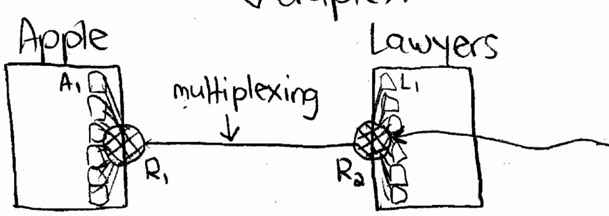
Internet
~ service view: delivers bits from one place to another
~ formal definition: set of all reachable IP addresses
~ topology view
Protocol
~ formal relationship and interaction



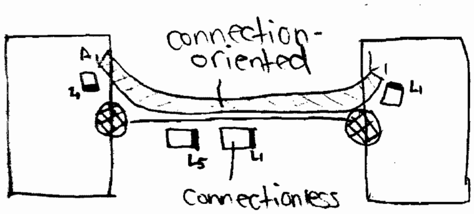
05 Jan 2012



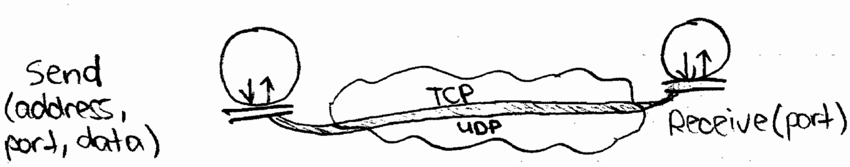
simplex: ⟶
half-duplex: ⟶ or ⟵
duplex: ⇄



~ addresses
~ port numbers
demultiplexing



datagram
connection-oriented: cell network
connectionless: internet

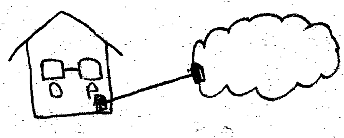


IP Address
a.b.c.d
0-255

IP

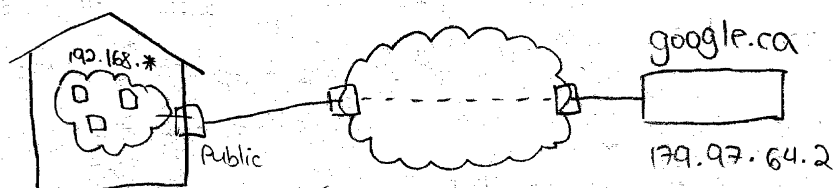
~ Public : reachable

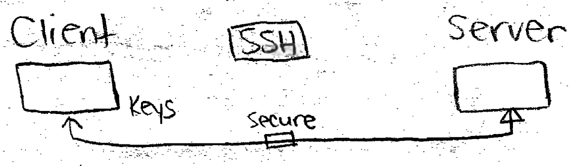
~ Private : 10.*.*.* , 192.168.*.*



D = desktop
P = printer (private)

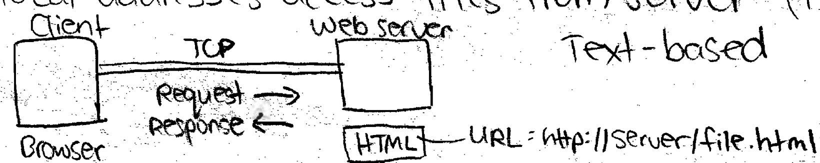
Network Address Translation (NAT)





NFS : local addresses access files from server (remote)

WWW :



Text-based

$\underbrace{\text{GET}}_{\text{method}}$ $\underbrace{\text{/somedir/somefile.html}}_{\text{location}}$ $\underbrace{\text{HTTP/1.1}}_{\text{version}}$

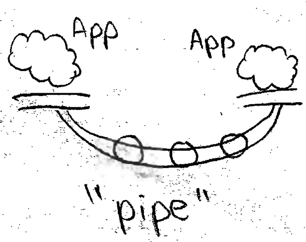
Host : URL

User-Agent : Mozilla

Accept-Language : en

~ cookie : string file (like an ID)

10 Jan 2012



1. Reliability
   ~ Error rate
   - bit error rate
   - packet loss rate

e.g. send "1111" using redundancy : (1111)(1111)(1111)

e.g. require confirmation for each packet

2. Throughput ("bandwidth" - incorrect use of word)

~ bits/sec

3. Delay
   ~ speed of light propogation delay
   ~ queueing
   ~ link throughput

~ speed of light takes about 1/4 of a second to travel around the world, 4-80 min for mars

~ link throughput:

10^6 bits

10^3 bits/sec

10^3 sec $\approx$ 20 min

limited by file size

4. Security/Privacy
5. Cost
   ~ money
   ~ energy

~ how do these things interact?

Cookies

Client ———— Server

→ Request

Response ←

~ Stateless: no memory of any clients

~ Apache: "a patchy" server, stateless

Client ———— Server

→ Request (ID)

← Response (cookie)

Storage

db (cookie, URL)

Amazon

Doubleclick

→ Request (cookie)

← Customized response

Third Party Cookies

Google (Doubleclick)

~ knows what you bought on Amazon and watched on CNN

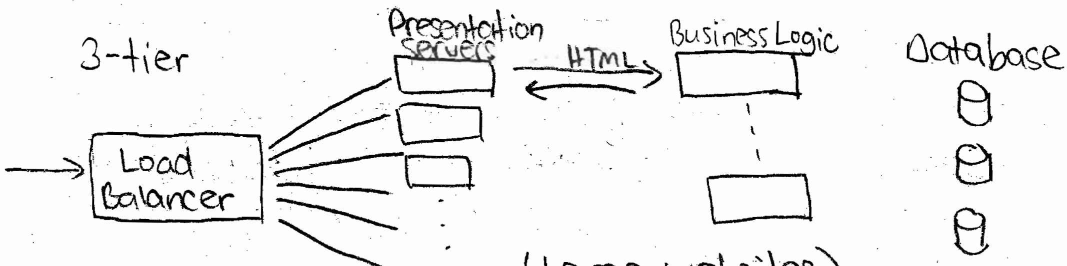
CNN.com

~ Another method: use ads to track your behaviour

Web

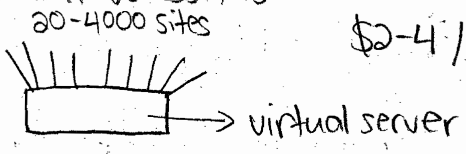
~99% of all hits for top 1000 websites

~ 3-tier



(Large websites)

~ Small websites:



\$2-4 /month/page = \$thousands profit
e.g. 1&1 (Germany)

DNS

~ giving a baby a name based on coordinates and time
~ or, ... grandparent . parent . baby

Registrar
serverlol. cs. uwat. ca (tree)
unique
top level domain
Domain Name Space (DNS)

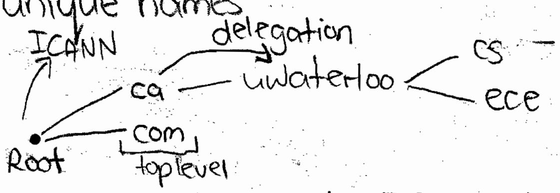
~ hierarchy
~ delegation
~ namespace
DNS Name ⟶ IP Address

1. Easier to remember names
2. Load Balancing
3. Aliasing (name to name)

12 Jan 2012

Domain name service:

1. unique names

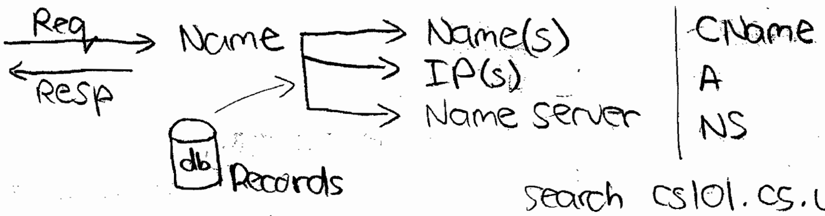


2. Mapping names to IP addresses
3. Aliasing: Names ⟶ Names
eg. uwaterloo.ca ⟶ www1.uwaterloo.ca ⟶ IP, IP, IP

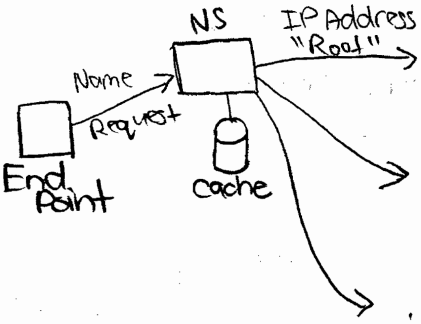
officemax.com → staples.com

uwaterloo.ca → { www1.uwaterloo.ca, www2.uwaterloo.ca, ⋮ www9.uwaterloo.ca }

Name Server:



search cs101.cs.uwaterloo.ca



Root
NS CA NS1.ca
A NS1.ca 128.32.6.4

NS1.ca
NS uwaterloo.ca NS1.uwaterloo.ca
A ns1.uwaterloo.ca 129.97.128.2

...

(Application layer)

"resolution" name → IP, iterative or recursive

Iterative: Your local NS keeps requesting to other servers

Recursive: Each request goes deeper into the servers

NSLookup and Dig

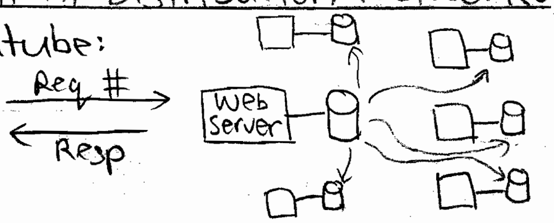
Better than having one centralized server:
- if an owner changes their IP, difficult/hassle
- speed of light limit
- very heavy load on one server

TTL: Time to Live (86400 seconds = 1 day)

Story: Who is the Donkey?

Content Distribution Networks (CDN)

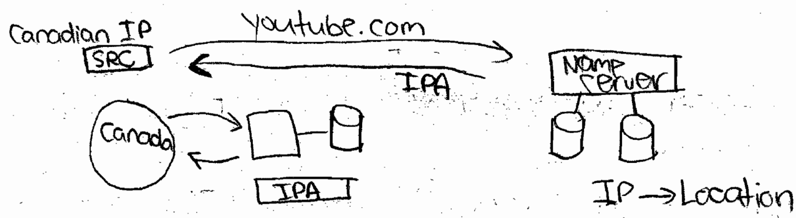
Youtube:



Push and pull content distribution

each "copy" server has a cache which stores the most recently requested videos

Request routing: detects your location and sends you video from geographically nearby server



~ Akamai : how to route requests – "akamization"

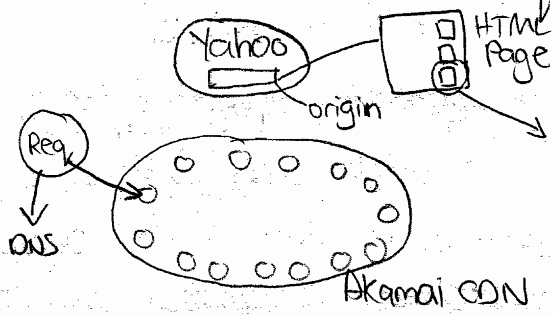
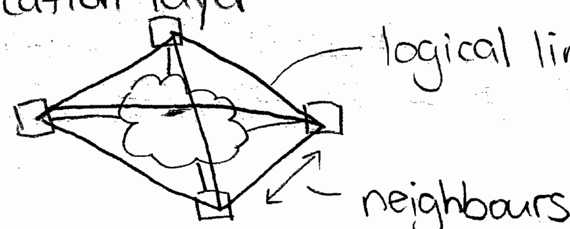


Image tag
URL → Akamai's name
Yahoo.com/img/img1.jpg to
yahoo.akamai.com/img/img1.jpg

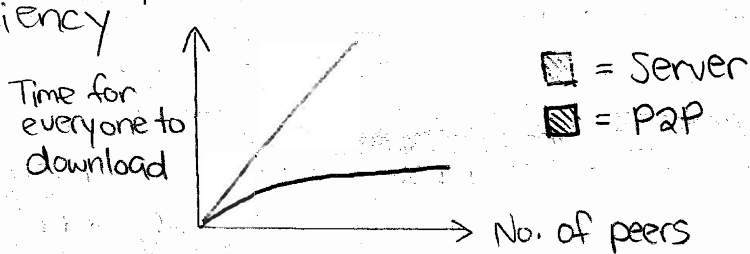
~ Akamaization : change the image tag
~ CName aliasing : images.yahoo.com yahoo.akamai.com

17 Jan 2012

<u>Peer-to-Peer</u>
~ Goal of P2P : efficient use of resources
~ application layer

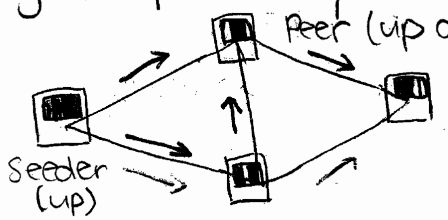


~ multiple copies of content
~ anonyminity
~ efficiency



<u>BitTorrent</u> (2001)
~ Torrent :
- key to overlay network
- description of file

~ Trackers:
- keep track of peers in overlay: IP Address, sharing stats
- e.g. thepiratebay

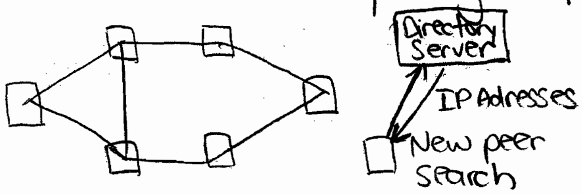


Chunks: ~256 KB

~ if leecher isn't sharing with a peer, that peer will drop the connection

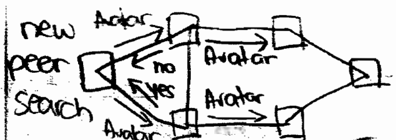
~ freeriding: connecting to new peers when being dropped

~ Search:
- torrent search engine
- centralized directory (e.g. Napster)



Advantage: No 3rd party search, easier to implement
Disadv: 1 point of failure, hard to scale

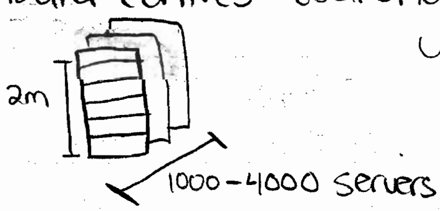
- decentralized search



e.g. Skype
~ in data centres (e.g. facebook bittorrent)

The Cloud

~ a datacentre
~ centralized content
~ online apps.
~ renting computing power
~ Physical view:
- Data centres: warehouse full of servers



usually >10 000 servers
Microsoft: 200 000
Cooling: ~ major cost
~ in cool regions

~ Service view:
- e.g. gmail, dropbox, youtube

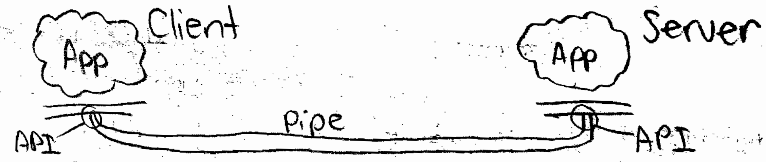
Applications
Platforms
Infrastructure
Virtualization
Servers + Hard Disks

Platforms (PaaS)
e.g. Windows Azure
~auto scaling ~reliability ~data backup
Infrastructure (IaaS)
e.g. Amazon Web Services
~rent a server ~more flexible

Virtualization: Allows you to "slice" a server

~ Advantage of Cloud:
- users can access data from anywhere
- stable, "unlimited" power
- flexibility
- much cheaper cost (e.g. on demand, per use)

19 Jan 2012



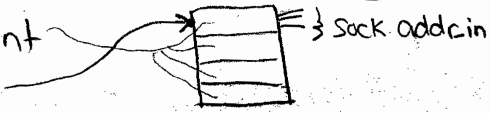
~ Prof K's Tattoo Parlor

<u>Client</u> | <u>Server</u>

Establish "store front" socket()
Listen bind()

Determine server location
Connect ⟶
Accept
R/W (Socket) ⟵⟶ (Socket) R/W
Close | Close

~ Socket FD = int
Socket Table



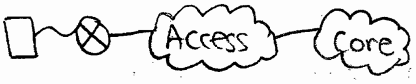
Sock Addr_in
- Address family: AF_INet/(AF_Uni
- IP Address: IP of server or "any"
- Port number: Server's Port

Host Ent - what is returned by DNS

~ Story: Eye of the bird
~ refer to printed program

24 Jan 2012

Application	Email, HTTP, SSH, P2P, DNS
Transport	Goal: <u>Logical</u> communication between apps. Apps think they're ~~directly~~ connected.
Network	Routing datagrams between networks
Link	Communication between nodes on same link
Physical	Air, copper, fiber

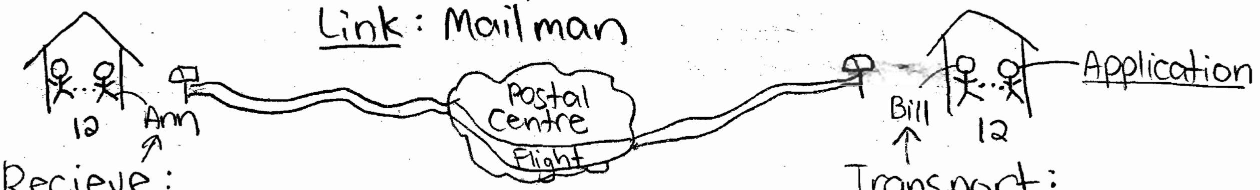


Vancouver

<u>Network Layer</u>: Canada Post

<u>Link</u>: Mailman

KW



Recieve:

1. Collect from mailbox
2. Hand letters out.

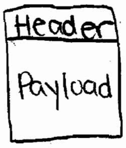
<u>Physical</u>:

Trucks, Planes

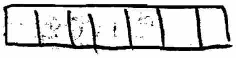
<u>Transport</u>:

1. Collect
2. Take to mailbox

Header: Info attached to a packet to indicate destination, src (e.g. envelope)



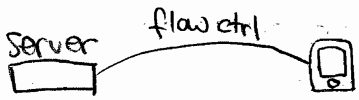
Segment: Transport-layer piece of the msg

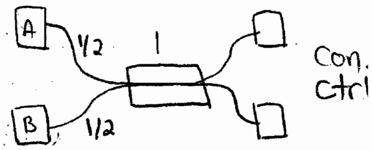


Network provides <u>best</u> <u>effort</u> service

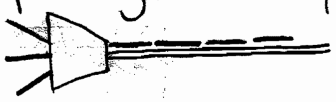
Transport can provide services

- reliable delivery
- flow control
- Congestion control



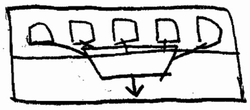


Multiplexing / Demultiplexing



→ Multiplexing: Many signals to 1 medium

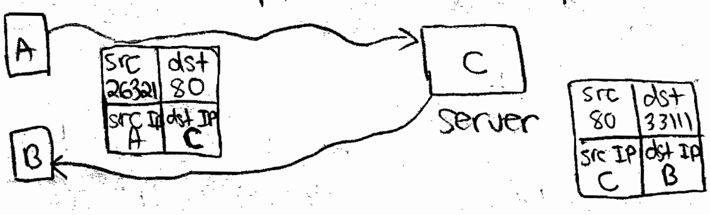
← Demultiplexing: Multiple signals sharing channel, separates



Applications

Transport

~ Ports
- unique identifier of applications on end-host
- 16 bits, 2^{16}
- 0 – 1023 reserved for well-known services
- Port 80: http, Port 21: ftp



UDP

~ User Datagram Protocol
~ Goal: As simple transport as possible
~ Connectionless: as soon as there's data to send, send it
~ UDP header: 16 bits

src port (optional)	dst port (required)
Length (of data)	CheckSum (optional)
Data ⋮	

header

Error detection if some no. of bits are corrupt, then transport protocol (UDP) detects this

~ UDP lacks:
- reliability
- in order delivery
- flow control
- congestion control

TCP (1974)
- complicated protocol

Why UDP?
- faster, fine-grained control, real-time apps, no connection stat
- no connection establishment, small packet header, VOIP, video chatting, some video games

~ What protocols are used:

	App	Transfer
Email	SMTP	TCP
Web	HTTP	TCP
File transfer	FTP	TCP
Remote file servers	NFS	UDP (typically)
Streaming	proprietary	UDP
VOIP	proprietary	UDP
Network Management	SNMP	UDP

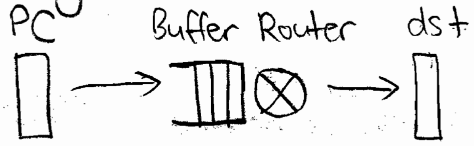
26 Jan 2012

Buffer:

~ Bucket that holds packets until a system is ready for them.

Reliability

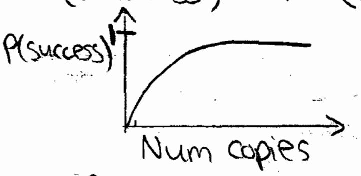
~ Messages sent will be delivered

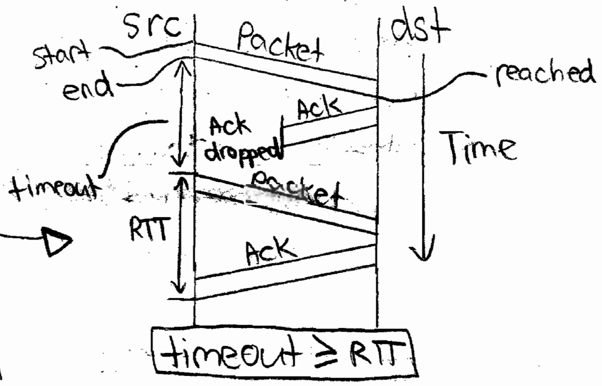


If buffer is full, packet is thrown away.

~ Multiple copies

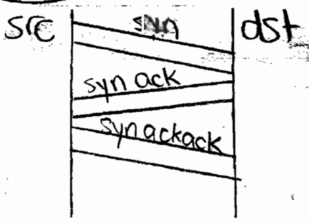
$P(\text{success}) = 1 - (\text{error rate})^{\text{Num of copies}}$

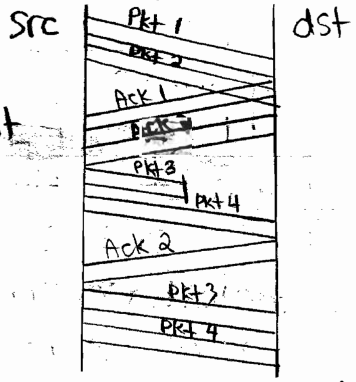




~ Ask for acknowledgement

- Timeout = some amount of time, after which we assume packet was dropped
- Round trip time = time for a packet to be sent to dst and for dst to acknowledge
- Cumulative ack
- 3-way handshake:

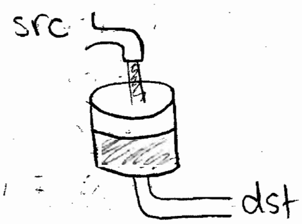




Flow Control

~ Techniques to match a src's sending rate to the service rate of the dst and network devices between src and dst

~ Objectives:

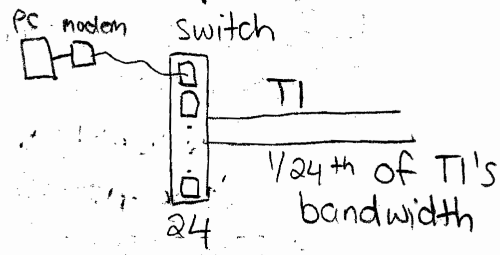


- simplicity
- efficiency (minimal overheads)
- fairness (bandwidth share)
- stability (converge to an equilibrium)

Open Loop Flow Control

~ No feedback between sender and dst

~ Admission control
- user asks for certain requirements (descriptor) like bandwidth, latency
- dst admits / rejects request
- eg. dial-up:

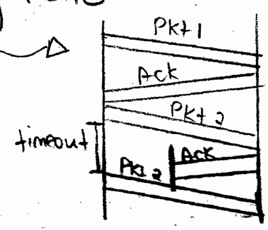


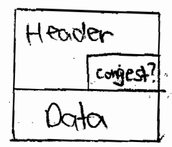
<u>Closed Loop Flow Control</u>

~ Sender and dst dynamically decide on a sending rate

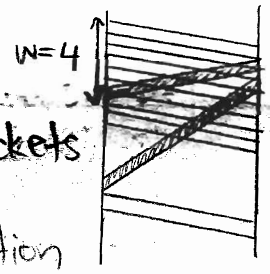
~ Devices do <u>not</u> reserve enough bandwidth for communication (statistical multiplexing)

~ Explicit: Network devices use explicit control messages to negotiate the sending rate
- e.g. Stop and wait

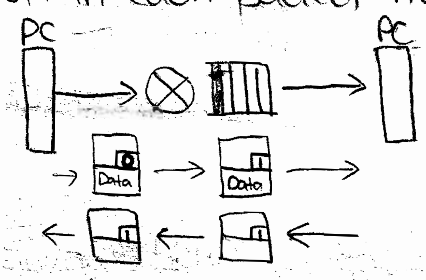




~ Static window:
Send w packets before waiting,
w = transmission window = # un-acked packets



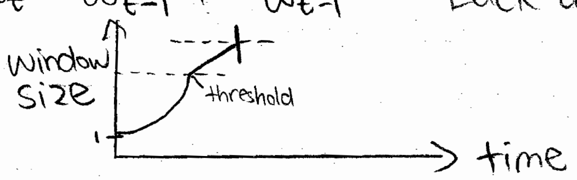
~ <u>Explicit</u> congestion notification
- bit in each packet header indicates congestion



0 : no congestion
1 : congested

~ Implicit flow control: Dynamically adjust transmission window, in response to undelivered packets

~ Dynamic w:
- start at 1
- Each ack, double the window size until w reaches a threshold (slow start)
- $w_t = w_{t-1} + \frac{}{w_{t-1}}$ Each ack, increase window



31 Jan 2012

~ Flow control : match sending and receiving rate
~ Segment : part of a message
~ Buffer : bucket that holds segments until sys is ready
~ Open-loop flow control : no feedback
~ Admission control : only allow access from connections we know we can support
~ Closed-loop flow control : feedback, statistical multiplexing
~ Ack : acknowledgement that packet was recieved
~ Timeout : after this amount of time, we assume pkt lost
~ Sequence numbers : No. assigned to each pkt

<u>TCP</u>

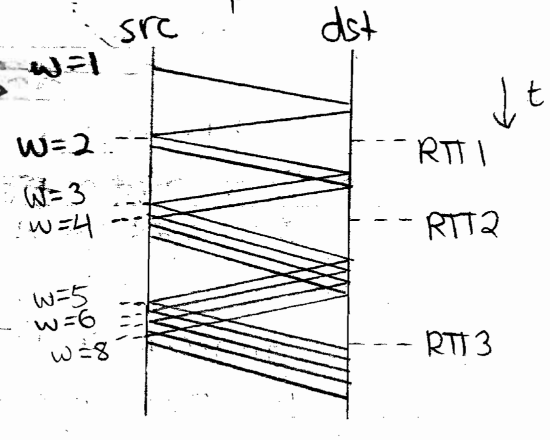
~ Reliability
~ Flow control
~ Congestion control

} Same mechanism

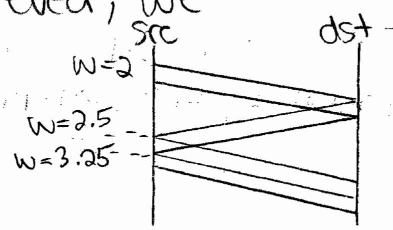
Only info is : Did the packet reach dst?

~ Transmission window W : No. of un-ack'd packets we're allowed to send

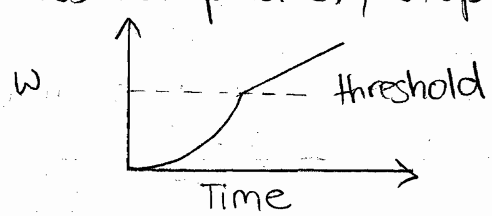
~ **Dynamic window : init w=1**
Slow start : each time recieve ack, increase w by 1



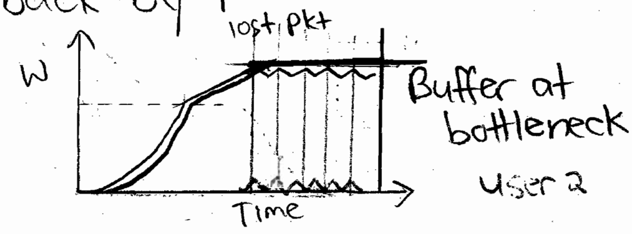
~ Congestion avoidance :
When ack receieved, we set $w = w + \frac{1}{w}$



When there is a loss of packet, step w back by 1



Normal

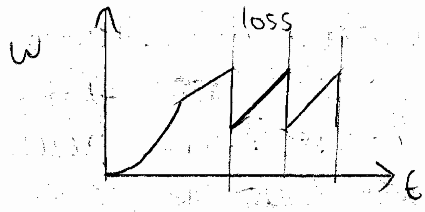


Packet loss

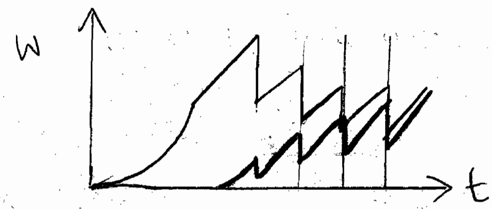
Unfair

Fairness: n senders, $\frac{1}{n}$ transmission bandwidth

~ Additive increase, Multiplicative decrease (AIMD): when loss detected, cut w in half



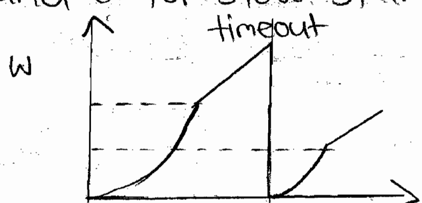
1 sender



2 senders

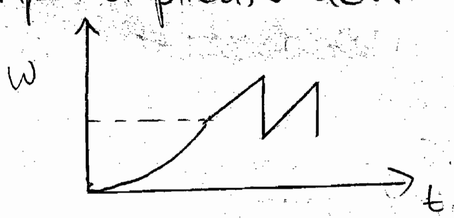
~ Timeout: Go back to pkt with sequence no. that was not ack'd, resend this packet, and following pkts. Set w=1 and enter slow start

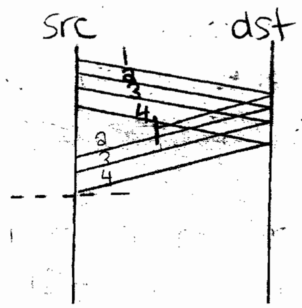
W=1



Threshold is cut in half

~ Triple duplicate ack





w is cut in half. Threshold = new w, stay in congestion avoid

TCP Details:

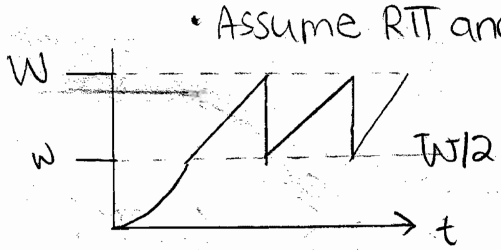
~ Close connection: End-host A sends FIN. End-host B ACK and sends FIN. A ACK's the FIN.

~ How to set timeout for packet loss? Measure the RTT timeout $\geq$ RTT + something to account for variance

~ Average throughput: (throughput = transmission rate)

$\frac{w}{RTT}$ W = w when packet loss occurs,

- Assume loss always detected by triple dup. ack.
- Assume RTT and W are const for the duration



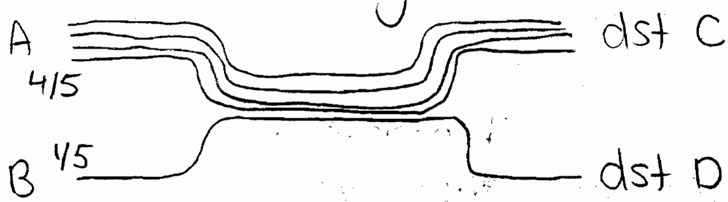
Average throughput $= \frac{3/4\ W}{RTT}$

02 Feb 2012

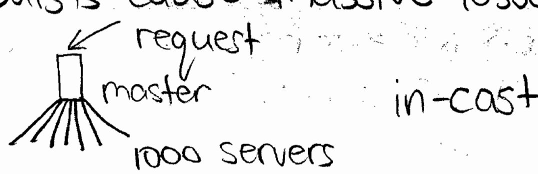
- Flow control: Regulate the sending rate to match the receiving rate vs. Congestion Control: Control the entry of traffic into the network to prevent congestion collapse
- Timeout: Time until we declare a packet lost
- Throughput $= \frac{W}{RTT}$

<u>Drawbacks of TCP</u>

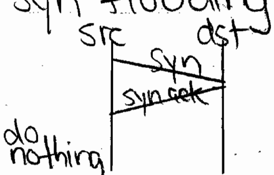
- Fairness can be gamed



- Assumes that packet losses mean congestion
  e.g. 10 Gbps link, 1500 byte packets. In order to send at 10 Gbps: $W = 83\,333$. 1 packet loss every 5 billion packets. Unrealistic!
- Detects congestion using loss
  - TCP has to fill the bottleneck buffer
  - Bursts cause massive losses



- Sensitive to threshold
  e.g. If $W=1$, 3 RTT's to get a webpage, but if you start with $W=10$, only 1 RTT needed
- Syn flooding

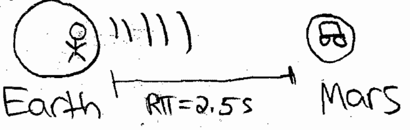


  - a form of Denial of Service Attack (DoS)
- Only info TCP has is packet loss
  - have to overflow buffers to find amount of available bandwidth

~ Explicit congestion notification
- when buffer nearly full, we mark congestion field
- drawback: all network devices have to support it
- Implicit: Changing protocol only involves changing end-hosts

<u>What is an Ideal Transport Protocol?</u>

~ Depends on the network



100 Mbps

100 Mbps · 2.5s = 256Mb = <u>32 MB</u>

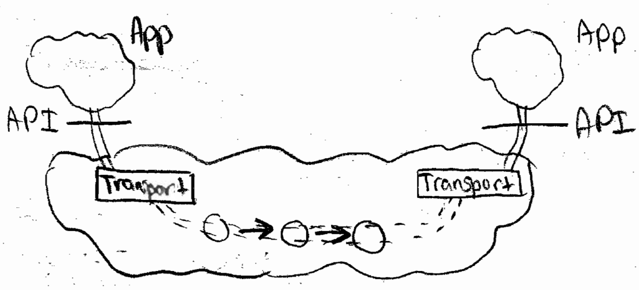
before receiving an ack

~ Datacentre Networks:
- high bandwidth, low RTT
- Microsoft: TCP + explicit congestion notification (ECN)
- single owner of datacentre
- buy switches with ECN support
- ECN: Additive decrease in window size

~ Wireless Networks
- High loss rates
- Solution: Use link layer to help TCP
- ∴ Links themselves help TCP by quickly retransmitting lost segments

~ TCP works very well on wide range of networks
- Wide Area Networks (the Internet)
- works well for wide range of RTT's
- widely tested
- best we have

07 Feb 2012



~ addresses

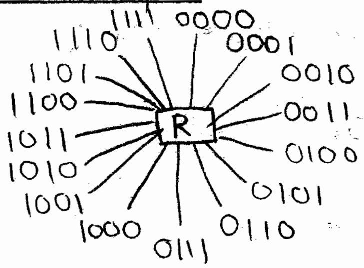
~ forwarding

~ routing

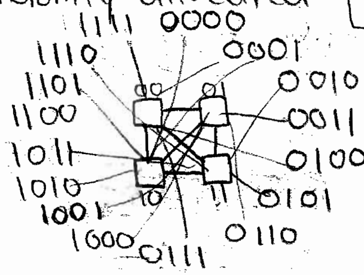
~ geo. addressing

~ hierarchy

Network Layer



Randomly allocated:

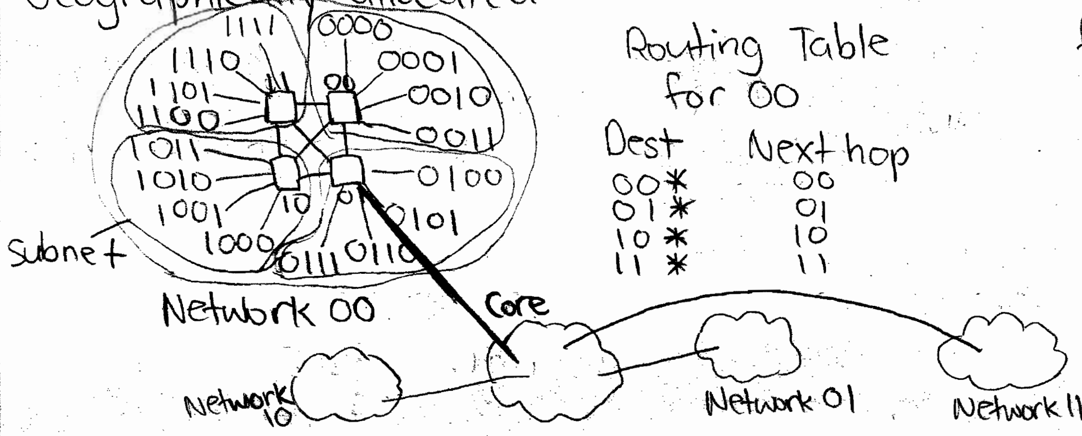


Routing Table for 00

Dest	Next hop
0011	01
0111	10
⋮	⋮

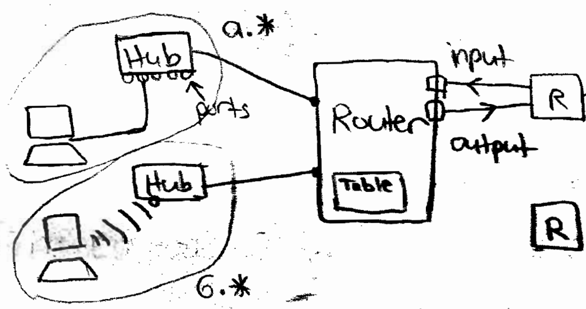
12 entries for each table

Geographically allocated:

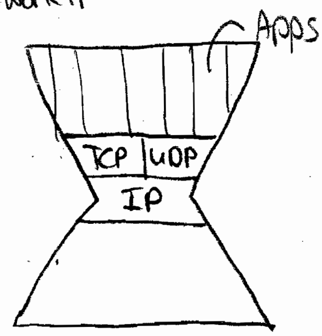


Routing Table for 00

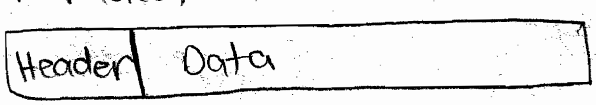
Dest	Next hop
00*	00
01*	01
10*	10
11*	11



IP
Internet
Protocol



- IP Addresses: IPv4 - 32 bits, a.b.c.d., $a,b,c,d \in [0,255]$
- e.g. 129.97.|24.|2   129.97.0.0/16
  Netmask   First 16 bits relevant
- ARIN allocates netmasks
- 3 layers: core, network, subnet
- IP Packet



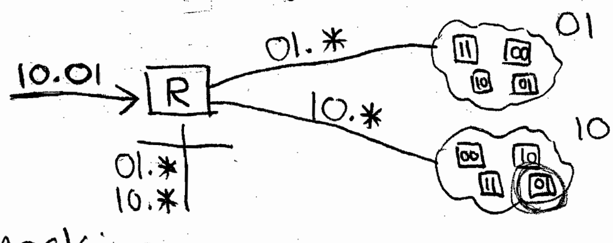
↳ version, src addr, dst addr

09 Feb 2012

- 32 bit IP Addresses, $2^{32} \approx 4$ billion IP's

~ Aggregation

$\underbrace{a.b.c.0}$ $\underbrace{/30}_{\text{mask}}$ ignore 32-30 = last 2



~ Masking

$$\begin{array}{lll} & 1011\,0111 & /4 \\ \text{AND} & 1111\,0000 & \leftarrow \text{mask} \\ \hline & 1011\,0000 & \end{array}$$

select top 4 bits

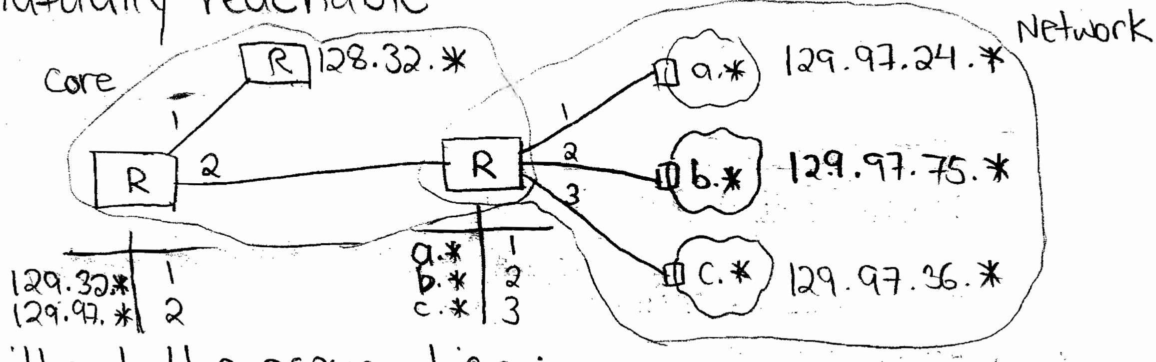
e.g. 129.97.24.* is a /24 network

~

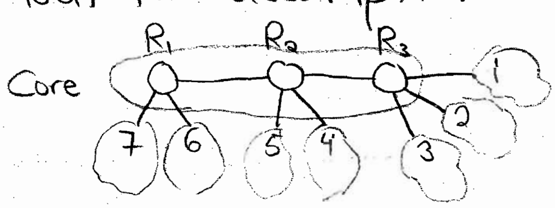
129.97.75.0	129.97.75.4
129.97.75.1	129.97.75.5
129.97.75.2	129.97.75.6
129.97.75.3	129.97.75.7
129.97.75.0 /30	129.97.75.4 /30

~ Assumption: All addresses within a subnet are mutually reachable

~



~ Without the assumption:



R_1:

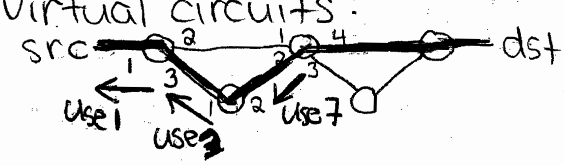
Dst	Next
4	R_2
5	R_2
1	R_2
2	R_2
3	R_2

R_2:

Dst	Next
1	R_3
2	R_3
3	R_3
6	R_1
7	R_1

~ Datagrams = Postcards

~ Virtual circuits:



- pin paths
- use short addresses

Step 1: Send datagram

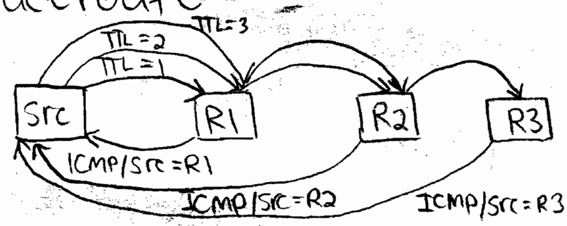
Dst	Port	Next label
1	3	3 (from "use 3")

14 Feb 2012

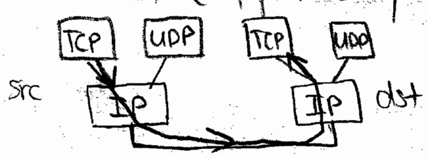
IP Header

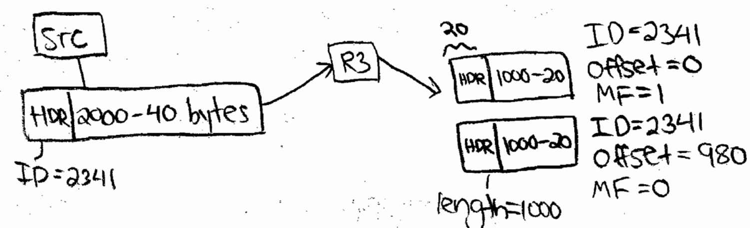
1. Version < 4 - 32 bit, most used; 6 - bit, "the future"
2. Header length (in "words" - 4 bytes) ~ 20 bytes (5 words)
3. Options
4. Type of Service < Delay-sensitive (e.g. Skype); Delay-insensitive (e.g. bitTorrent)
   - Not used because can't be enforced
   - ATM service, but was dropped
5. Length (header + Data) in bytes : 16 bit (65536 bytes)
6. TTL : Time to Live : set to 0-255 by source. Decremented by every router. if field == 0 → drop. Source is sent an error message - Internet Control Message Protocol (ICMP) - Packet expired

~ Traceroute



7. Protocol (Upper Layer)





8. ID
9. Fragment Offset
10. Fragment Flags < DF: Don't frag; MF: More frag

- Grouped together
- spoof by setting all MF=1

11. Header checksum (sum = 65536)

~ Parity bit:

- Even parity: 1011 011 $\boxed{1}$ 0111 100 $\boxed{0}$
- chosen so sum always adds to particular number (65536

12. Src address

13. Dst address

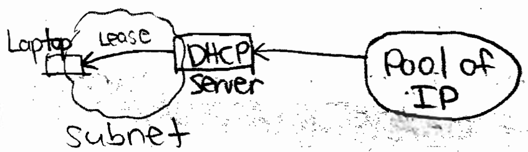
<u>Address Allocation</u>

~ IP Address

- For an interface

~ 2 steps to get an address:

1. Allocation: Organizations get a block of addresses
   - Internet Assigned Numbers Authority (IANA). Registrars, e.g. ARIN (North America), RIPE (Europe)
2. Hand it out (DHCP): Dynamic Host Config Protocol



Lan or subnet:

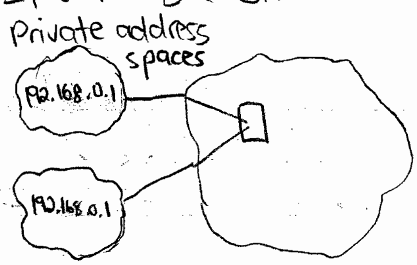
Media ~~Access Address~~ (MAC)

special address - broadcast, subnet number 111111

16 Feb 2012

~ IPv4: 32 bit

Private address spaces



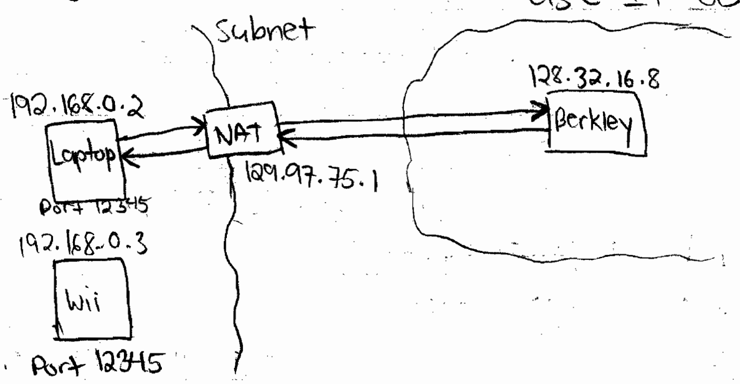
"net ten"

10.* /8 - first network (ARPANET)

192.168.0.0/16

- need NAT to translate (Network Address Translation)
- use IP as you wish

~



GNAT (TCP/IP) Orig Port#

Src	Port #	dst	Orig Port
192.168.0.2	12001	128.32.16.8	12345
192.168.0.3	12002	128.32.16.8	12345

~ timeout for NAT table ≈ 30 min

~ (funny network story)

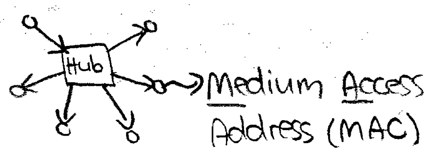
~ App Layer
Transport – TCP/UDP
Network – IP
Link Layer – Ethernet
Physical

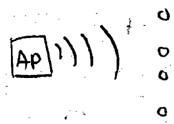
↔ Duplex

→ or ← Simplex

o——o point-to-point

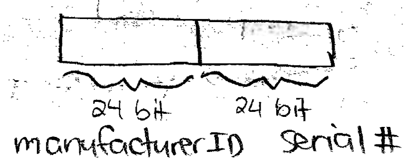
~ Broadcast medium





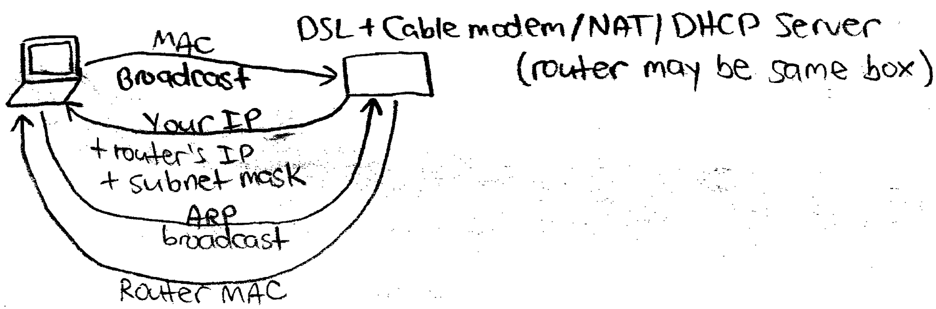
1. How do interfaces get a unique MAC address?
2. How can a source learn the destination address?

~ Ethernet MAC – wired and wireless (wifi)

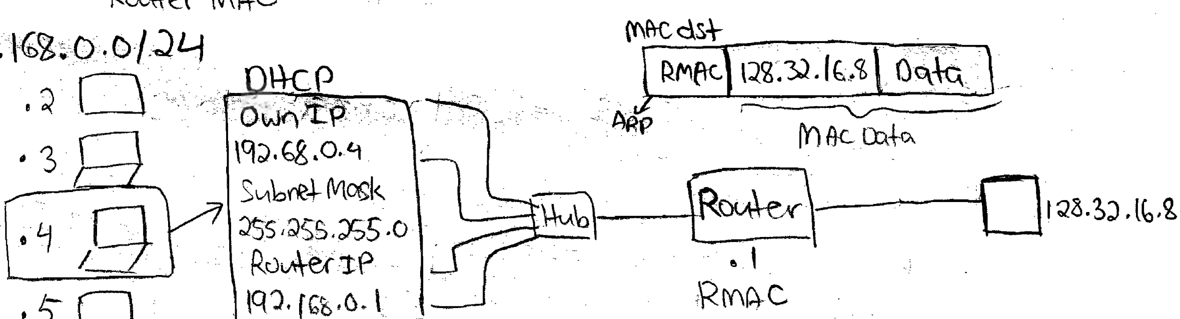


- get from IEEE
  $1000 per manufacturer ID

~ ARP: Address Resolution Protocol



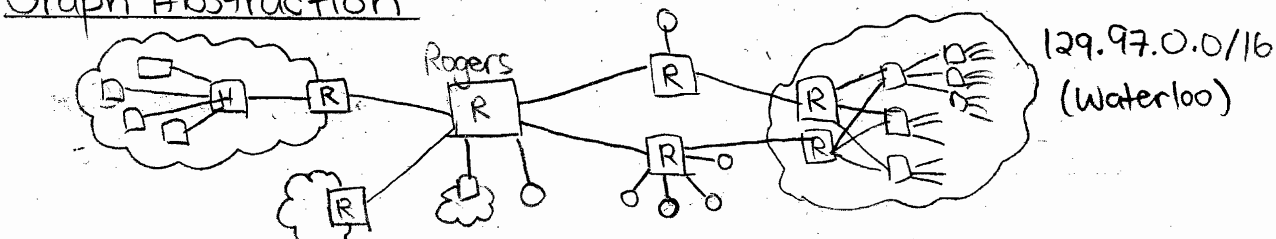
~ 192.168.0.0/24

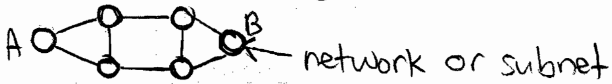


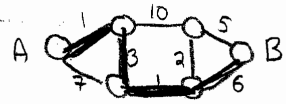
If ((dst IP And Subnet Mask) == (My IP And Subnet Mask))
Then Local → get dst MAC from ARP
Else Remote → dst MAC = RMAC

~ 255.255.255.0 means the same as 192.168.0.0/24

<u>Graph Abstraction</u>

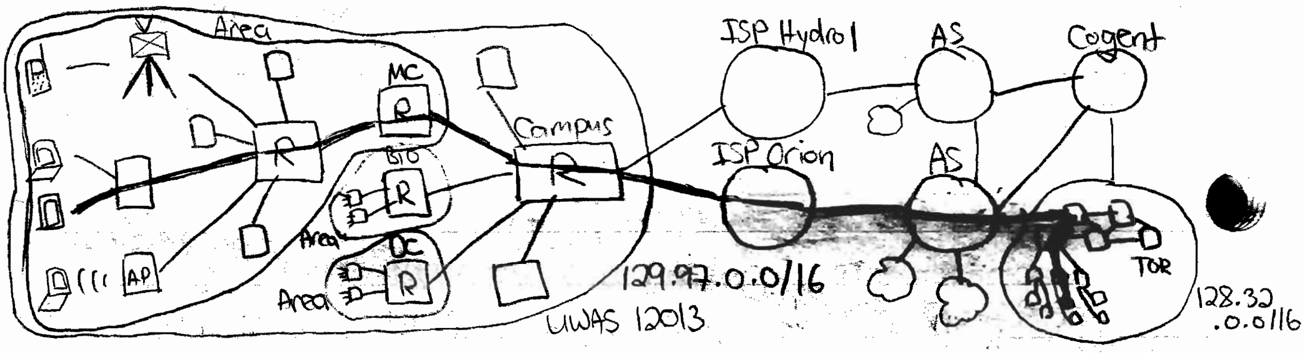


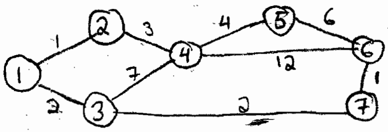




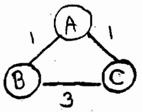
weight represents "goodness of link"
low weights are better

28 Feb 2012





Higher weight is worse



Two algorithms:
- Link state
- Distance vector

Link state - Assume:

1. Each node knows its attached edges and the nodes at the other ends
2. Links are bi-directional
3. Link weights are shared in common
4. All node names are global

A: | A | B | 1 |

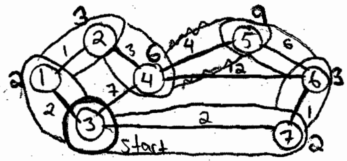
| A | C | 1 |

C: | C | A | 1 |

| C | B | 3 |

B: (own, nbr, wt) | B | A | 1 | , | B | C | 3 |

Link state packets
- flooded, rarely
- everyone gets packet
- travels each edge once



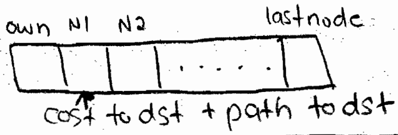
"growing blob"
Dijkstra's algorithm

Node	time	path
3	0	self
1	2	3-1
7	2	3-7
6	3	3-7-6
2	3	3-1-2
4	6	3-1-2-4
5	9	3-7-6-5

OSPF
Open Shortest Path First

~ Happiness = Have − Want

~ Distance Vector



| A | B 1 | C 1 |

| B | A 1 | C 3 |

| C | A 1 | B 3 |

• sent only to neighbours

What B Knows

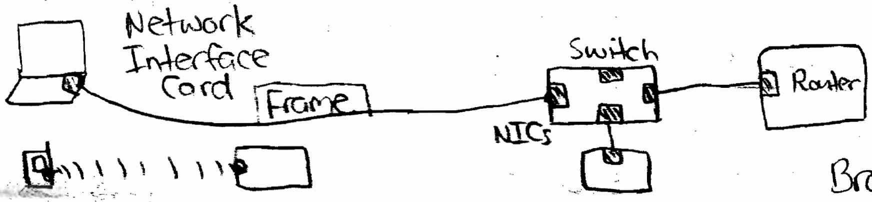
Time	Node	Cost
0	A	1
0	C	3
Receives packet from A		
1	C	2 through A

BGP - Border Gateway Protocol

01 Mar 2012

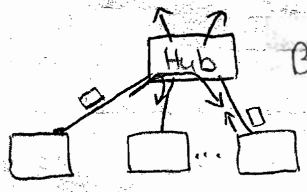
Link Layer

~

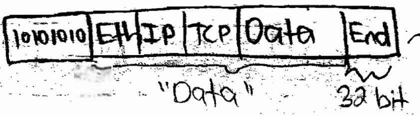


Point to Point

Broadcast < wireless, Hubs



Broadcast
• collisions



~ Cyclic Redundancy Checksum (group theory)

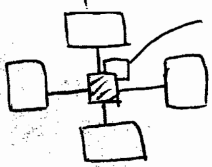
~ Services
• Framing
• Error control:

Integer + x % 7 = 0

Checksum
3,4
14,0
23,5

~ App: x, Trans: segment, Net: packet, Link: frame, Physical: bits
• Flow control: say "stop" if buffer full

~ Multiple Access



controller < msg, time

• frequency division multiplexing (muxing) → allocate freq
• time division multiplexing → time base
• code division multiplexing → allocate codes

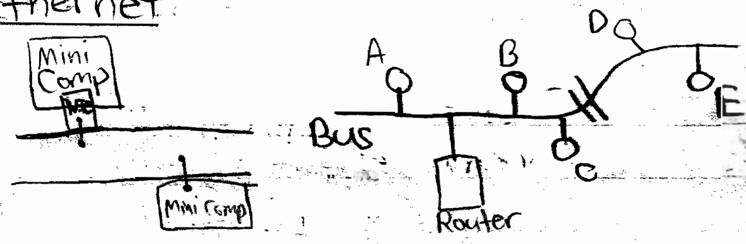
• polling
• token ring
• aloha

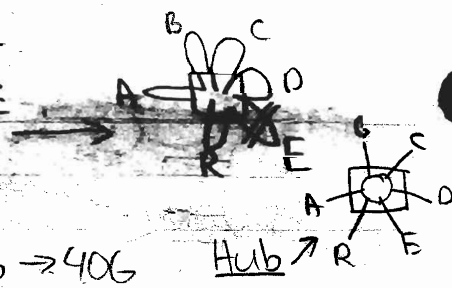
- CSMA – carrier sense multiple access
- CSMA/CD – CSMA + collision detect + random backoff
  Used by Ethernet
  If collision
      choose backoff_value
          random (1, backoff)
          backoff = 2 * backoff
      count down backoff value
  Try again

~ Freq + Time : cell phones (Rogers)
~ Code : cell phones (Bell)
~ Aloha : cable
~ Story : Metcalf (Ethernet)

Ethernet

~





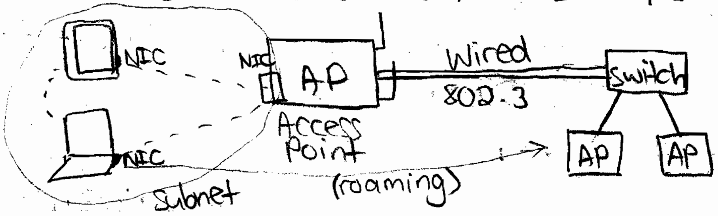
- 1 Mbps → 10 → 100 → 1G → 10G → 40G
- copper wire → fiber
- hub → Ethernet switch (no CSMA needed)

~

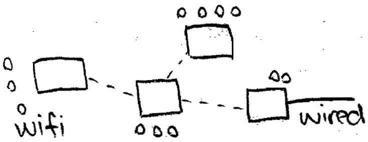
Preamble	Start of Frame	MAC Dest	MAC SRC	Type	IP TCP Payload	Data load	CRC
7 * 10101010 64 bits	10101011	6 bytes	6 bytes	what kind of eth upper layer			32 bit
		3 manufacturer, 3 serial #					

06 Mar 2012

~ Ethernet : 802.3
Wifi : 802.11 (b, a, g, n, ac)
~ Standard distribution : IEEE/IETF/ITU

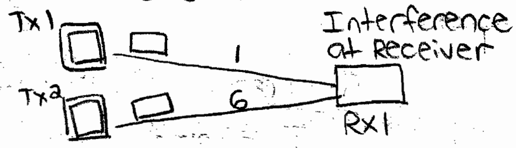


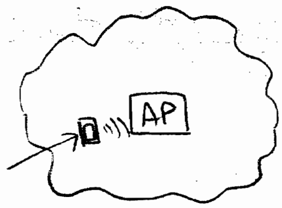
- infrastructure mode
- ad hoc mode

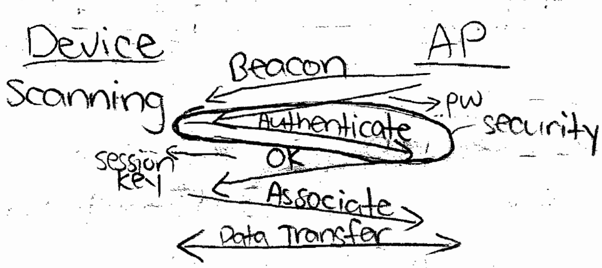
~ 

wireless mesh
- messy
- doesn't work

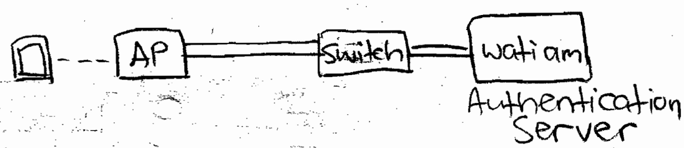
~ Wireless "Link"
- noisy
- channels
- interference at receiver due to collision

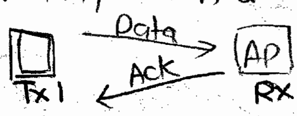


~ 802.11 b : 1, 2, 11 Mbps, 3 channels (1, 6, 11), 2.4 GHz
~ 802.11 a : up to 54 Mbps, 12 channels, 5GHz
~ 2.4 GHz covers more/better than 5GHz
~ 5 GHz is absorbed by water/people, 700 MHz is better but used by television
~ 802.11 g : up to 54Mbps, 3 channels, 2.4 GHz
~ 802.11 n : up to 108 Mbps, 1 channel, 2.4 GHz
~ 802.11 ac (coming)
~ 



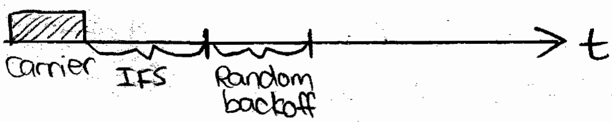
~ (cash register story)
~ Security
- over-the-air
- authentication
- Wired Equivalent Privacy (WEP) : easy to break
- WPA, WPA 2



~ 

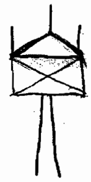
~ Wired vs Wireless (802.3 vs 802.11)
1. Ack
2. Random backoff <u>before</u> Tx
3. Double backoff on collision

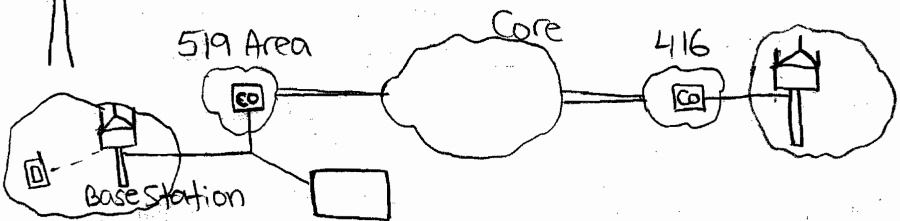
] CSMA/CA (collision avoidance)

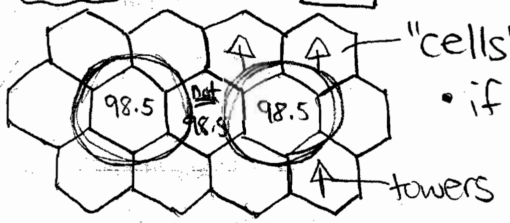


IFS : Interframe spacing

08 Mar 2012

~  cell phone tower

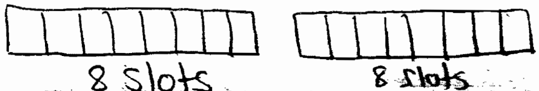


~ 

- if driving fast, handover

~ 1G - analog

~ 2G - FDM + TDM (GSM) - digital



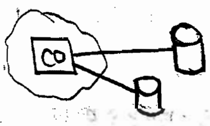
- time division
- frequency division

~ 3G - data-oriented - peak ~200 kbps

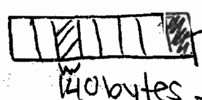
~ 4G - all IP (IPv6) - peak 1Gbps ↓ 500 Mbps ↑

~ LTE-Advanced, IMT-Advanced (4G)

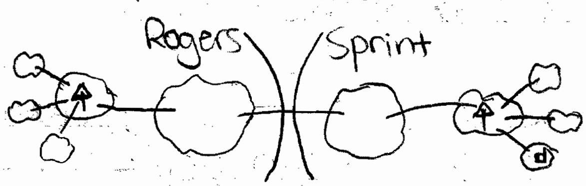
~  IMSI
(Sim Subscriber ID Module)



Home Location Register (#, credit card)
Visitor Location Register (#, cell tower)

~  signalling channel : when to ring, etc

SMS - Small Messaging Service

~ 

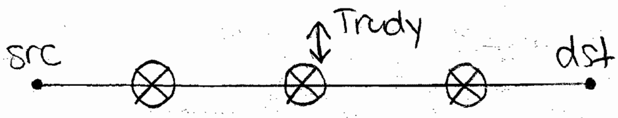
Roaming

~ OFDM

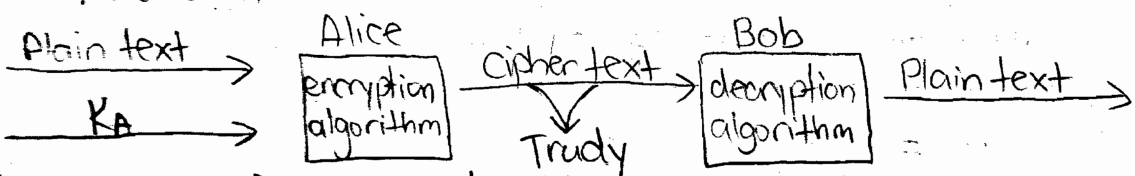
~ MIMO -
multiple in
multple out

4 paths
8 kbps → 1 Gbps

13 Mar 2012

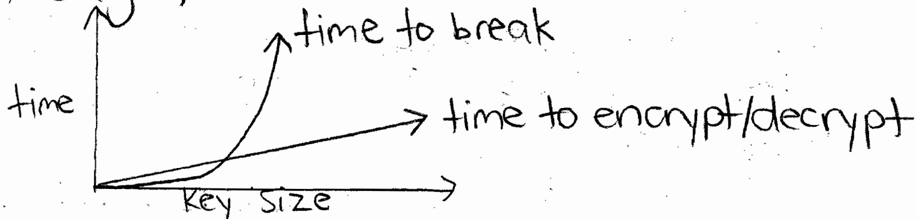


- Types of Attacks
  - copy
  - injection
  - replace / modify
  - spoofed
  - inferred
  - preventing delivery (DoS)
- Western Digital hack story
- Ken Thompson
  - put a backdoor in any UNIX system
  - no evidence in code
- Pentagon Tiger team
  - server in locked room
  - DEC letter forged work request
- House Keys
  - forging by taking a photo
- Despite the presence of malicious parties:
  - Privacy: Messages can't be eavsdropped or inferred
  - Authentication: Messages sent to right party
  - Integrity: Can't be tampered with
  - Denial of service: Ensure delivery
- Encryption
  - Encode a message so that only intended receiver can read it



- How secure is encryption?
  - attacker could try all keys
  - strength of encryption depends on number of keys

• breaking encryption : should be exponential in key strength



~ How practical?
• Usually depends on efficient encrypt / decrypt
• Security depends on long keys (and hard to guess)

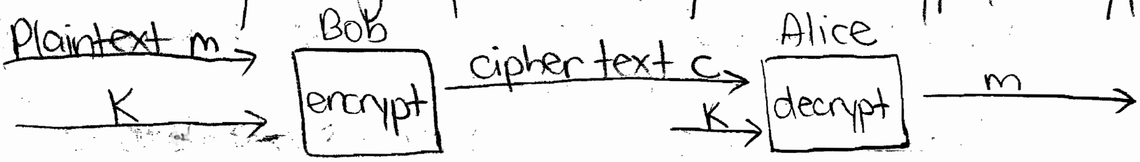
~ Time to check all 5-letter passwords (lower case)
• $26^5 \sim 10$ million
• In 1975 : 1 day, 1992 : 10 sec, 2008 : 0.001 sec

~ 6 letter password : upper, lower, numerical, control
• $70^6 \sim 600$ Billion
• In 1992 : 6 days, 2008 : < 1 sec using 1000 PC's

Cryptography

~ Two types : secret key (symmetric), public key (asymmetric)

~ Secret Key : single private key for encrypt / decrypt



~ First scheme : $a \rightarrow e$, $b \rightarrow f$, $c \rightarrow g$
• only 25 combinations (N-1)

~ Second attempt (random) :
$a \rightarrow d$; $b \rightarrow b$, $c \rightarrow a$, $d \rightarrow e$, $e \rightarrow d$, $f \rightarrow c$
• N! combinations (6! or 26!)

~ Both parties have to know key

~ One-time Pad :
• random sequence of bits that's the same length as m
• m = 01000110 XOR $\oplus$
p = 11011001
c = 10011111
• secure because m and p are same size (bits)
• 2^m combinations : too large! (size of msg)

~ DES (data encryption standard)

- key is 56 bits
- $m = \boxed{c_1 | c_2 | c_3 | c_4 | c_5 | c_6}$ Cipher block Chaining

$$\boxed{C_3 | C_1 | C_2 | C_5 | C_6 | C_4}$$

~ Public Key Crypto : keys for encryption are public (K_A^+, K_B^+), keys for decryption are private (K_A^-, K_B^-)

- Alice wants to send m to Bob
  $c = K_B^+(m)$
- Bob decrypts by
  $m = K_B^-(K_B^+(m))$
- Relies on the existance of one-way functions

~ RSA (Rivest, Shamir, Adleman, 1978)

- Choice of public/private keys
- encrypt/decrypt algorithms

~ Bob needs to choose K_B^+, K_B^-

1. Choose two large primes, p and q
2. Compute $n = pq$ and $z = (p-1)(q-1)$
3. Choose a number e, ($e < n$) and e and z are relatively prime (no common factors)
4. Find d such that $ed - 1$ is divisible by z
5. $K_B^+ = (n, e)$, $K_B^- = (n, d)$

~ Alice wants to send m to Bob ($m < n$)

- To encrypt she computes m^e and the remainder when m^e is divided by n: $c = m^e \bmod n$
- To decrypt, Bob computes $m = c^d \bmod n$

15 Mar 2012

$$\left.\begin{array}{l} 1.\ p, q \text{ primes} \\ 2.\ n = pq \text{ and } z = (p-1)(q-1) \\ 3.\ e \text{ st } e < n \text{ and } \gcd(e, z) = 1 \\ 4.\ d \text{ st } ed - 1 \equiv 0 \bmod z \\ 5.\ c = m^e \bmod n \text{ and } m = c^d \bmod n \end{array}\right\} \text{RSA}$$

~ e.g. Bob chooses $p=5$ and $q=7$. Then, $n=35$ and $z=24$.
Choose $e=5$, $d=29$. $a=1, b=2, c=3, \ldots$
Message = "L". $m=12$. $c = m^5 = 248832 \bmod 35 = 17$.
Decrypt: $m = 17^{29} \bmod 35 = 4819\ldots \bmod 35 = 12$. M = "L".

~ Why does RSA work?
$K_B^-(K_B^+(m)) = (m^e)^d \bmod n$.

~ Theorem: If p, q are prime and $n=pq$, then $x^y \bmod n = x^{(y \bmod (p-1)(q-1))}$, so:

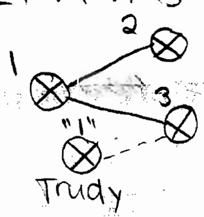
$$K_B^-(K_B^+(m)) = m^{(ed \bmod (p-1)(q-1))} \bmod n$$
$$= m^1 \bmod n \quad (\text{b/c } \bmod (p-1)(q-1) = 1)$$
$$= m \quad (\text{since } m < n)$$

~ Breaking RSA is as hard as factoring
~ Quantum computers: poly-time factoring
~ RSA is compute expensive
~ DES is much faster
~ Use RSA to exchange private keys then switch to DES.

Message Integrity

~ To authenticate a message, Bob must verify:
- Message originated from Alice
- m was not tampered with in transit.

If m has these properties, it has integrity.



← Routers. Trudy poses as Router 1.

Cryptographic hash function:
- computationally infeasible to find two messages x and y st. $H(x) = H(y)$.
- e.g. MD5 (Rivest), SHA-1

~ First attempt at message integrity:
1. Alice has m, computes $H(m)$
2. Sends $(m, H(m))$ to Bob.
3. Bob computes $H(m)$. If $H(m) = h$, everything is OK.

Attack: Trudy creates m', sends Bob $(m', H(m'))$

~ We need a secret!
~ Let authentication key = s

~ Integrity protocol:

1. Alice computes $\boxed{H(m+s)}$ – Message authentication code
2. Send $(m, H(m+s))$ to Bob.
3. Bob checks if $H(m+s) = h$. If so, everything is OK.

<u>Digital Signature</u>

~ Attests that an entity owns something or agrees to its contents.

~ Needs to be: verifiable, non-forgeable

~ Bob's signature must be unique.

~ Public key crypto has unique private and public keys.

~ Bob's signature is $K_B^-(m)$. Why?

- If his public key is known, then anyone can check his signature.

~ Attack: Trudy announces K_B^+ as her public signature.

~ Public Key Certification: Certifies that a public key belongs to a specific entity.

~ Certification Authority (CA):

1. Verifies that an entity is who it says it is.
2. Issues certificate that binds the public key to entity.
   - 36 commercial CA's

<u>End-Point Authentication</u>

~ Process of proving your identity.

~ Authentication Protocol 1.0 (ap1.0)

- Says "I'm Alice"
- Attack: Trudy says "I'm Alice"

~ ap2.0:

- Alice says ("I'm Alice", IP address).
- Attack: Trudy spoofs Alice's IP

~ ap3.0

- key = ("I'm Alice", password)
- Attack: sniff network traffic for password

~ ap 3.1

- key = ("I'm Alice", encrypted password)
- Attack: Playback attack (resend encrypted password)

~ ap4.0: Need to verify password is fresh.

1. Alice says "I'm Alice" to Bob.
2. Chooses a one-time-use number R, sends R to Alice.
3. Alice encrypts R using $K_{AB}(R)$, sends to Bob.
4. Bob decrypts. If equals R, then it's Alice.

Security in Practice

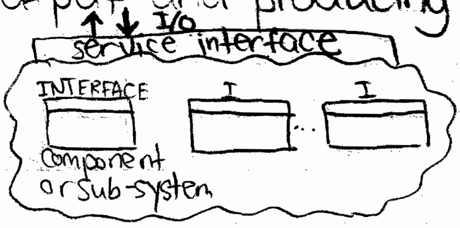
~ No way to prove secureness
- Can show known attacks don't work
- If no known attacks (long time), it should be secure

~ Black hat and white hat hackers
~ Vulnerability scanning
~ Password cracking
~ Packet sniffing
~ Spoofing
~ Root kit
~ Social Engineering *
~ Trojan horses, viruses, worms, keyloggers

20 Mar 2012

Distributed systems

~ System: Set of interacting components with specific input/output and producing functionality (service)

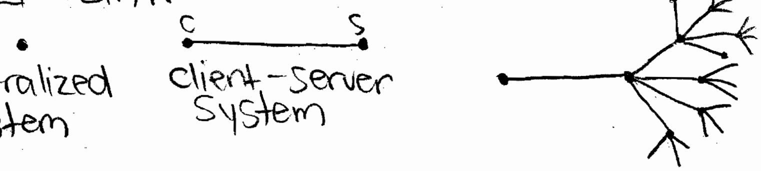


- computation
- storage
- communication

"resources"

~ SECURITY
Application → Distributed System (networked application)
Transport
Network
Link

~

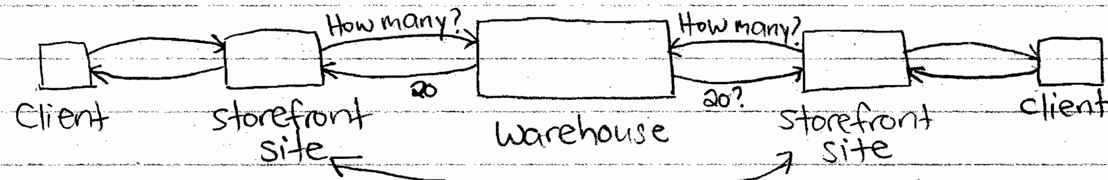


centralized System

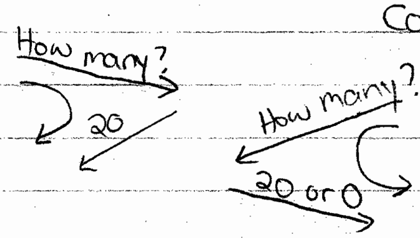
client-server System

Distributed System

CS 436 (cont'd)



Concurrency → inconsistensy



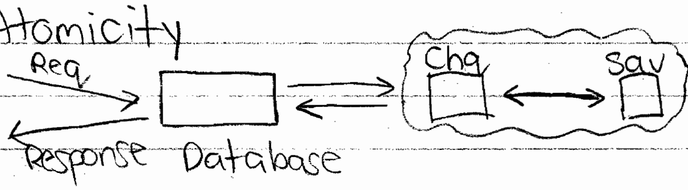
Desireable Properties
- atomicity
- consistency
- isolation
- durability

} ACID

- fault tolerance
- scalability
- synchrony
- transparency
- performance
- heterogeneity

~ Motivation:
- scalability
- fault tolerance
- delegation: local control
- localization

~ Atomicity



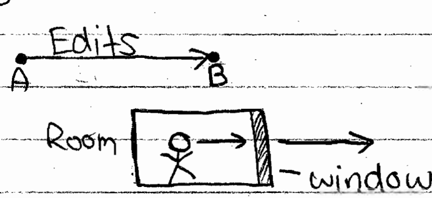
Transfer $200 from chequing to savings.

Read chq
Write sav
Write chq

} Transaction

~ Consistency: after transaction, chq amt + sav amt is the same.
~ Isolation: as if you're the only user on the system.
~ Durability: nothing gets undone after completed.
~ Fault tolerance: define faults
- Link failure, computer failure, disk failure
- Packet corruption, incorrect computation, disk corruption
- Combination of failures causes huge problems

~ Scalability: e.g. Wikipedia, google, cell phones
~ Performance: need a metric
~ Synchrony: "happens before"



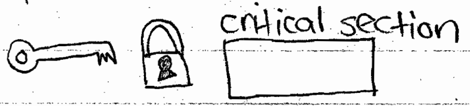
~ Transparency: services are independent of implementation

~ Heterogeneity: Different brands can be used

~ Locking (for Atomicity):

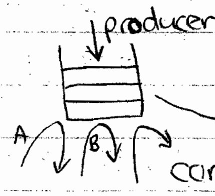
- concurrency + sharing

  processes   # of tic-tac-toe players



22 Mar 2012

- concurrent processes servicing the queue



~ if queue length > 0 then   (read)

  do work:

  - remove job from queue
  - decrement queue length   (write)
  - process job

  else wait for queue

- mutual exclusion
- need lock manager
- granularity < space / time

~ get_lock (ql, ID);

if (ql > 0) {   (read)

  remove job;

  ql--;   (write)

  release_lock (ql, ID); ← release as soon as possibly can

  process job;

} else release_lock (ql, ID);

~ Read Lock, Write Lock

- Many readers
- 1 writer + many readers

~ If program crashes before releasing lock, lock manager can check if it died - difficult to know what to do

~ Solution to dead lock: two-phase locking

~ Deadlock:

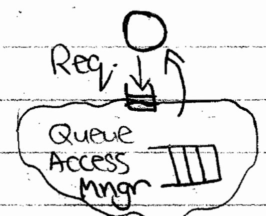
Process A	Process B
get_lock (ql1, ID)	get_lock (q2, ID)
get_lock (ql2, ID)	get_lock (q1, ID)

~ $A \rightleftarrows B$ $\quad$ $A \to B$, $B \to C$, $C \to A$

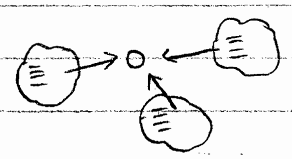
~ Solution : Lock ordering

~ Implementation :

1. Object-Synchronized : Java, C#, ...
2. Lock manager (pthreads) : C
3. Message passing



dequeue-queue

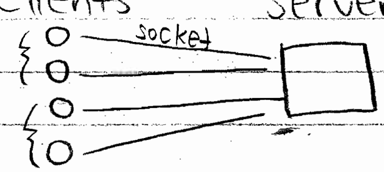


~ Design Principles

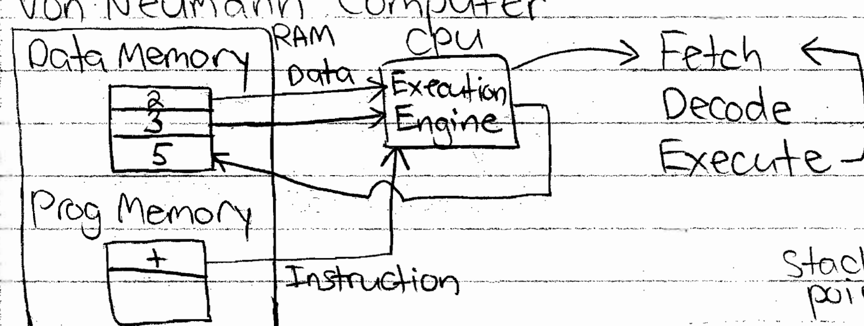
- No Global State
- No Central Control
- No Global Clock (?)
  GPS and Time server are solutions to this
- Separate Policy and Mechanism (customizability)
- Explicit Interfaces

27 Mar 2012

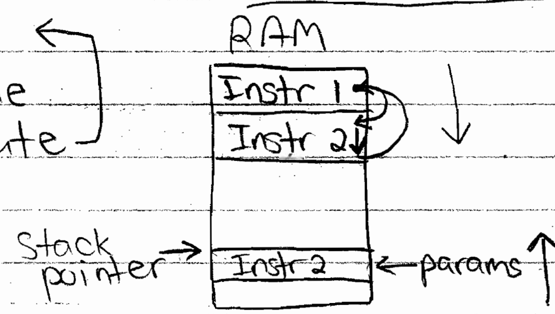
~ Client-server



~ von Neumann Computer



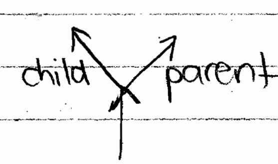
Procedure Call



~ Process :

- Instruction counter (PC)
- Register set (+SP)
- Memory state

fork()



~ Forking :

```
x = fork()
if (x == 0)        /* child */
   { do child stuff }
else
   { do parent stuff }
```

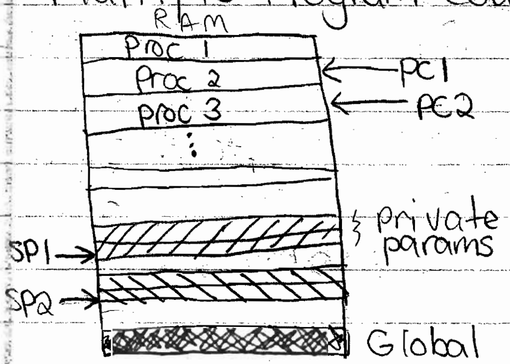
```
Parent

listen();
s = accept();
  x = fork();
  if (x == 0)
    {game} -> read/write (s)
else
```

~ Shared Memory needed for global variables

~ Threads

- multiple Program Counters and Stack Pointers



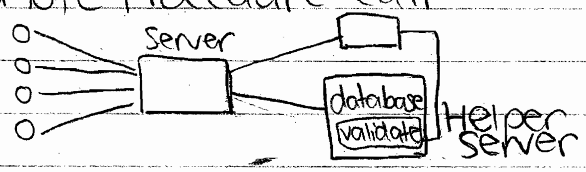
pthreads

- create thread (dummy)

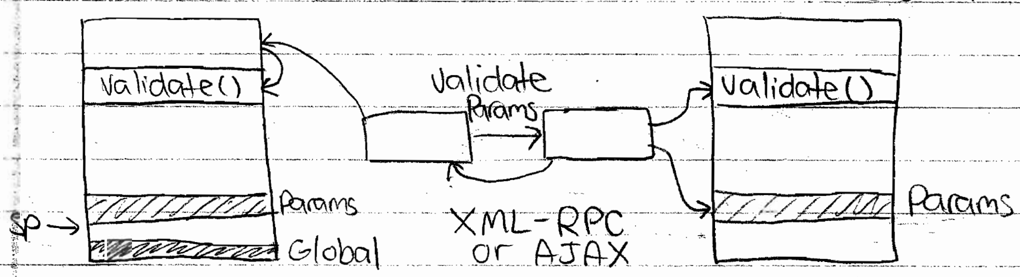
```
listen();
s1 = accept();    (np ++)
listen();
s2 = accept();    (np ++)
num_players += 2;
pthread_create(game, s1, s2)
```

- game() { ... np -= 2 }

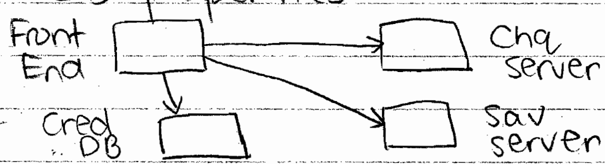
~ Remote Procedure Call




```
if (validate(login, pw))
   then ok
   else try again
```

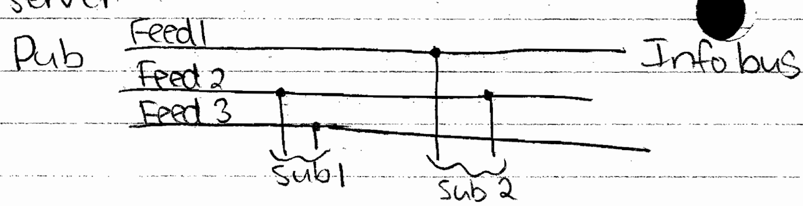


~ Transaction : Set of operations that satisfy the ACID properties



~ Publish subscribe

- Twitter feed
- RSS

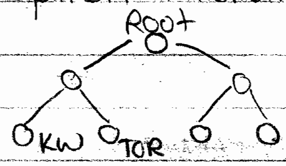


29 Mar 2012

Design Techniques

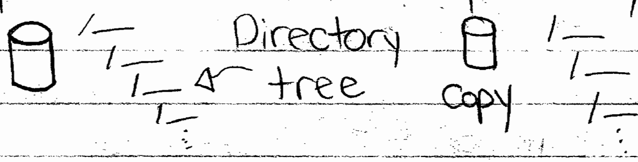
~ Hierarchy — Delegation

- DNS : server101.cs.uwaterloo.ca ⟶ IP Address
- Implicit Hierarchy : e.g. Yahoo Homepage



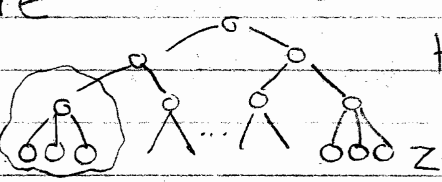
Delegation :
"Look for the name space"

~ Replication → Failover - primary / backup



Read-only is easy,
Read-write is very difficult

- Advantages : more capacity, scaling
- Disadvantages : cost, maintaining consistency, dealing with failure
- Google search :
  1 server
  x3 copies

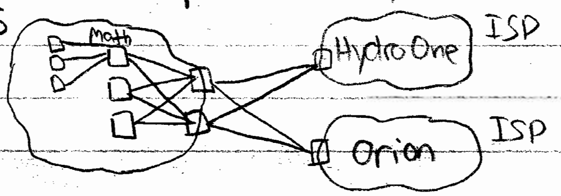


HDFS

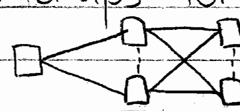
~ Story : "complience port" - Parnassus

- hot backup vs cold backup

UW AS



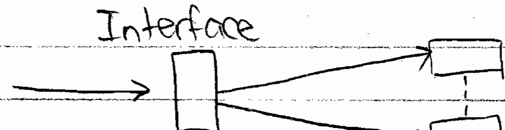
- both main routers are backups for each other



~ Indirection : "Go there"

Root Server :
IP addresses ⟶ .com
of TCP Name ⟶ .info
Servers ⟶ .net

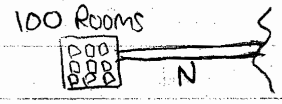
Interface



Front End
Load Balancing

- e.g. calling 1-800 #
- page tables

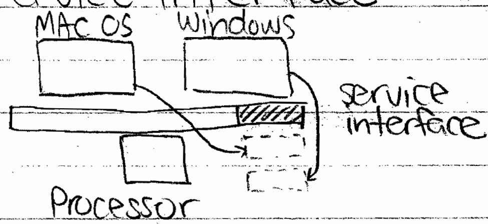
~ Multiplexing : "sharing"
- saves cost
- Demultiplexing : need ID
- performance can suffer

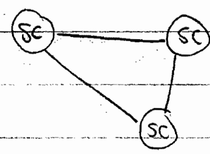


- N = ? Pick a number and adjust as it goes

~ 12 Phrases :
- Wake up
- Relax
- Breathe
- Feel the Earth
- Be grateful
- Be effortless
- Trust yourself
- Control your mind
- Do no harm
- Life is good
- Be happy now
- Let it go

~ Virtualization
- e.g. Santa Clause in a mall
- Service interface

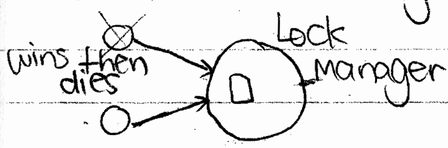




communicate to all santa clauses

- "private servers"

~ Soft State : "Forget it"



- have expiry time
- lock must be renewed
- DHCP lease

Some Exam Notes

Multiple Access
~ Frequency division: allocate frequency, "muxing"
~ Time division: time base
~ Code division: allocate codes
~ Polling: ask clients
~ Token ring: pass token
~ Aloha: send anytime, random backoff
~ CSMA: check before transmission, random backoff
~ CSMA/CD: CSMA with collision detect
~ CSMA/CA: require ack, random backoff before transmission, double backoff upon collision