Notes to go with CS 436, *Distributed Computer Systems*
These notes are only a supplement and are NOT a replacement for the textbook and/or other course notes that should be used to prepare for exams.

Tommy Carpenter
University of Waterloo

August 28, 2014

Contents

1 Brief Backround Needed for Rest Of Topics

1.1 What Is The Internet?

1. **Service View**: Bits being delivered to you.
2. **Topology View**: How do we build this system so it functions as above?
3. **Formal Definition**: *Set of all reachable IP addresses.* This means each persons view of the Internet is unique (the same set of IPs are not reachable from everywhere, e.g., firewalls).

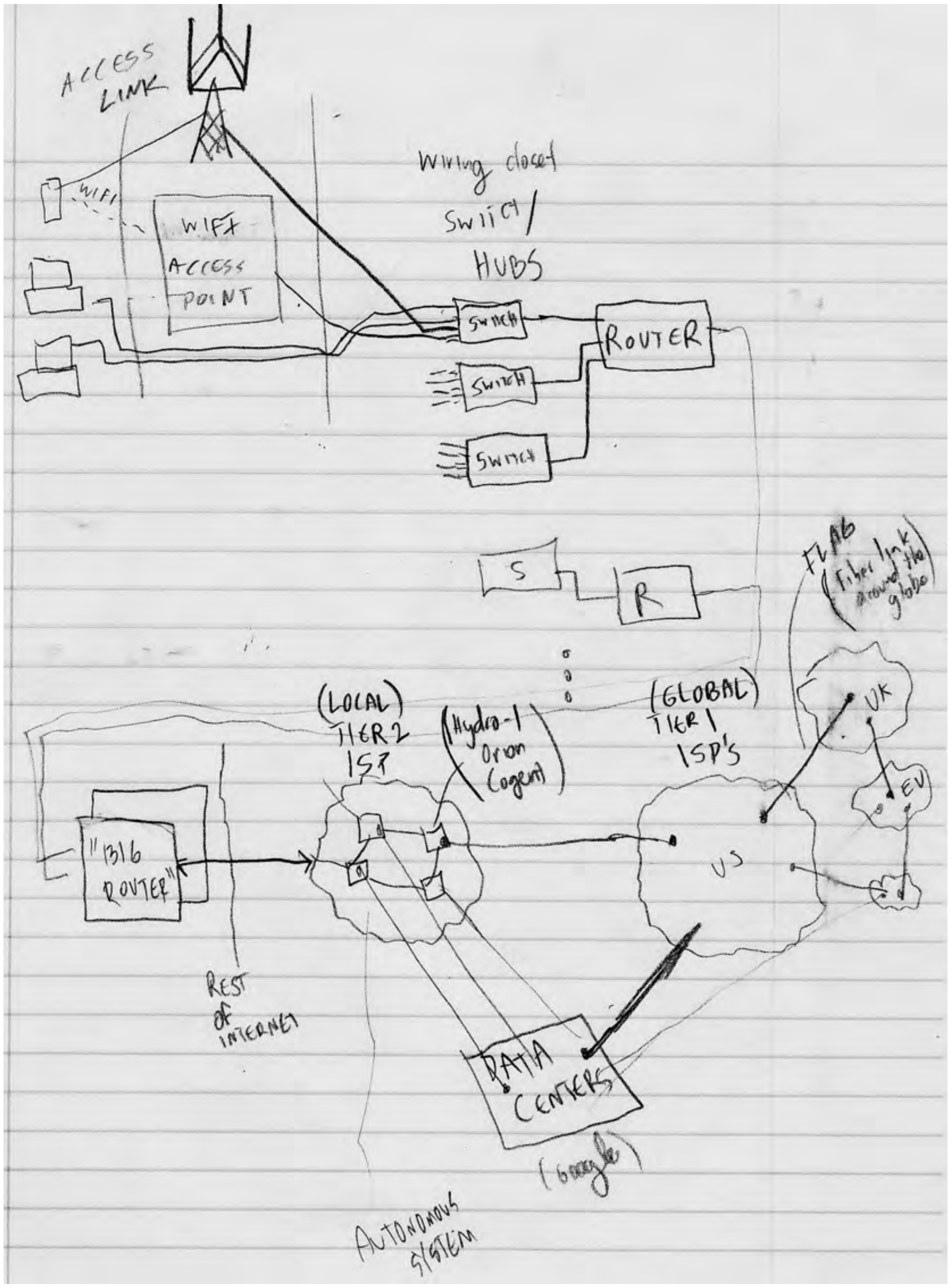


Figure 1: A View of Several Internet Components

1.2 Terms

- *Simplex*: one way channel only capable of one way communication
- *Half Duplex*: two way channel, but can only communicate in one direction at a time
- *Duplex*: two way channel capable of simultaneous two way communication
- *IP Address*: “Address” of one device connected to the internet.
- *Port Number*: Identifies an application inside one device (the device being addressed by its IP) . IP Address + Port Number uniquely identifies any application on any machine connected to the internet.
- *Connection oriented vs. Connectionless networks.* Example: circuit vs packet switched networks. In connection oriented networks, communication paths are established and are unshared end to end connections. No one else may use that path while the connection is established. Connectionless networks allow the sharing of paths by allowing multiple entities to send addressed information on the path.
- *Datagram*: chunk of information with a “header” that tells nodes along the path where it should be sent.
- *Protocol*: A way to formalize a relationship & interaction.
- *Buffers* store packets until a system is ready for them.

1.3 The Network Stack

top

1. Application
2. Transport
3. Network
4. Link
5. Physical

bottom

Analogy: There are two houses with a bunch of cousins that love sending letters to one another. One member of each household, Ann and Bill, is responsible for taking the letters from everyone inside their homes and bringing them to the mailbox. They are also responsible for collecting mail from the mailbox and distributing it to everyone inside the homes.

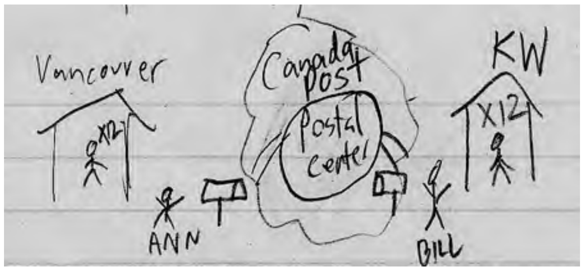


Figure 2: Picture of Analogy

- The children in the houses are representative of applications in the application layer. They expect their letters to be delivered to the other children but don’t care about any details of how this is done.
- Ann and Bill are representative of the transport layer. They take packets (letters in this example) from various applications and multiplex them into the network layer. They also wait on both ends of the network layer for messages to be delivered.
- Canada Post is representative of the network layer. They are the “cloud” between the two mailboxes.
- The path within the network layer from the two mailboxes to the postal sorting center can be seen as the link layer.
- Finally the trucks, roads, mailmen etc. are representative of the physical layer.

Note that Ann and Bill (transport layer) are constrained by the quality of Canada Post. If that service is unreliable, it is up to Ann and Bill to ensure reliability by implementing a scheme themselves. The network provides *best effort service*: it will try to deliver your message but makes no guarantees.

2 Application Layer

2.1 Main Purpose

Consists of applications that make use of connections between various networked end devices.

2.2 SSH

Secure (encrypted) connection between client and server. For more info, see http://en.wikipedia.org/wiki/Secure_Shell

2.3 NFS

Suppose we want a filesystem that can be accessed by different users across a variety of machines. This is the purpose of NFS. We can have directories on users machines that are actually links to directories on another machine. Files are accessed via the network, but this is abstracted from applications; they simply see a directory with files. The UW math computing environment is an example. When you log in, the *load balancer* sends you to one of many servers, yet you can access your files on all of them and could not tell the difference between being on one machine of another. They are not replicating your files across all the machines, instead they are just using NFS and give you networked access to files that are actually stored somewhere else. In short its a way to read, open, write files that are on a different machine; for more info see http://en.wikipedia.org/wiki/Network_File_System

2.4 WWW

Text Based System
Clients communicate with web servers. Requests (for HTML files) come from clients, responses come from servers.
HTML files contain data and links (Uniform Recourse Locators: URLs) to other files on other servers like http://server/file.html

2.4.1 HTTP

- HTTP structure: CLIENT → request → SERVER → response → CLIENT → repeat..
- Transcript of an example HTTP Client Server Interaction Session:
  GET /somedir/somefile.html HTTP/1.1 Host: URL User-Agent: Mozilla, Accept-Language: en
  HTTP/1.1 200 OK Content-type:img or HTTP/1.1 400 Not Found

2.4.2 Cookies

- The web was designed to be stateless, but this property has slowly degraded.
- Cookies are a way to track people.
- CLIENT sends REQUEST to SERVER. SERVER sends a cookie back as part of the HTTP response and contains information about CLIENT and its' location.
- The next time CLIENT sends a request to SERVER, it sends the cookie as well. SERVER then uses the cookie to identify you.

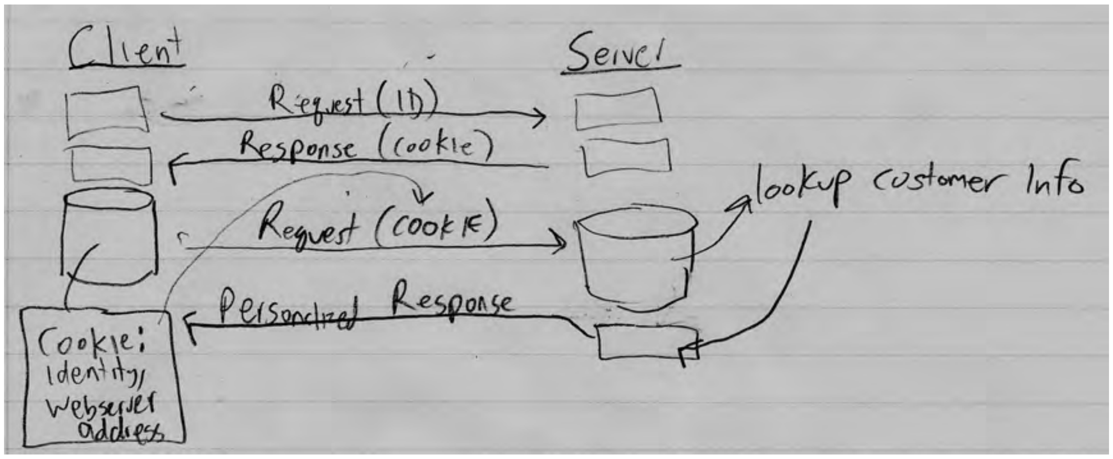


Figure 3: Cookie Transaction

Third party cookies can be used to track you. There are two ways TPCs are used to help companies sell products to you.

1. Product marketing. Amazon starts offering to sell other sites' cookies. When you go to Amazon, a cookie for not just Amazon comes back, but also an additional cookie for "amazonpartner.com". Amazonpartner.com now knows who you are and what you bought from Amazon, which helps them target products to you if you ever go to their site.
2. Personalized adds. Amazon has some add for "double-click.com". When you go to Amazon, they tell your browser "go to double-click and fetch this ad". Then your browser goes to "double-click.com", fetches the add, and receives a cookie back. From then on, whenever you go to any site that has a double-click ad, they can target the ads. Your browser now sends your already existing cookie to double-click so they have information on what you have purchased from sites with the double-click cookie and they also know what types of sites you have visited (they can see all the sites you have visited with double-click cookies).

2.4.3 Webservers

The BIG guys use a three tier architecture.
request → load balancer → *presentation tier* → *business logic tier* → *database tier* → (then back)

- Incoming requests go to a load balancer, hardware that sends your request to one of many *presentation servers.*
- The presentation servers assemble your HTML request ("HTML assembly"). First they send info to a *business logic server.*
- The business logic server talks to various databases at the *database tier* to assemble ads, track cookies etc. The databases store information about your cookies etc.
- After receiving the necessary info back from the business logic server, the presentation server then assembles the final HTML page and sends it back to you.

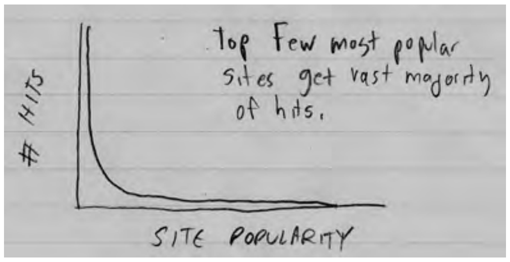


Figure 4: The number of hits per site is exponentially decreasing. The top few sites get the majority of all site hits.

2.5 Domain Name Space (DNS)

- Ensures unique names by using a dotted namespace.
- Top Level Domain: .ca,.com,.edu,.net, .xxx. ICANN is the TLD registrar.
- Consider the name x.y.z: there is a registrar for x, for y, and a registrar for z. Each registrar (for example the top level domain registrar) is responsible for ensuring all names in their namespace are unique - e.g., there is only one ".com". They are only responsible for their level and trust that other registrars also enforce uniqueness in their level.
- Each registrar *delegates* to the prior tier uniqueness responsibility. "I ensure uwaterloo.ca is unique in the .ca space. It is YOUR responsibility to ensure all x.uwaterloo.ca are unique."

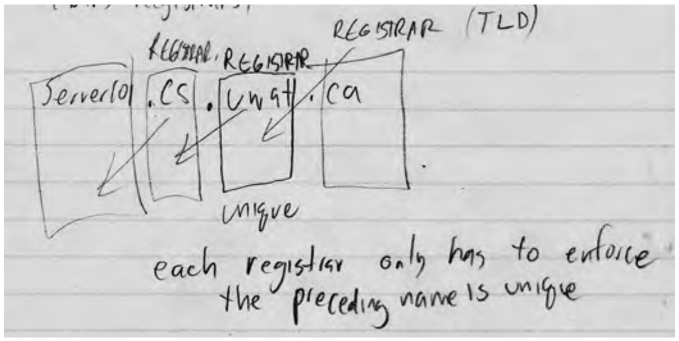

REGISTRAR (TLD)
REGISTRAR
REGISTRAR
Server10
.cs
.uwat
.ca
unique
each registrar only has to enforce the preceding name is unique

Figure 5: Registrar Delegation

Why do we use DNS instead of just using unique IP addresses?

- Human readability. When humans want to go Google.ca for example, they don't want to type 74.125.226.152 because that is impossible to remember. Instead, they want to type "google.ca".
- Aliasing (Names → Names). Examples:
  - uwaterloo.ca → {www1,www2,www3,...,www10}.uwaterloo.ca. In this case DNS does round robining for users; some users get directed to server 1, some get directed to server 2 etc. Sort of like load balancing via DNS.
  - Suppose X buys Y. When people go to Y.com, you want them to be rerouted to X.com
  - People move their websites. old.com → new.com

Possible DNS responses

1. CNAME: a record indicating your name was an alias, and here is the name it points to. Try that next
2. A: (Address) An IP address (your done, you have your IP)
3. NS: (Name Server) This name server can't help you, so here is a name of another name server. Try that next.

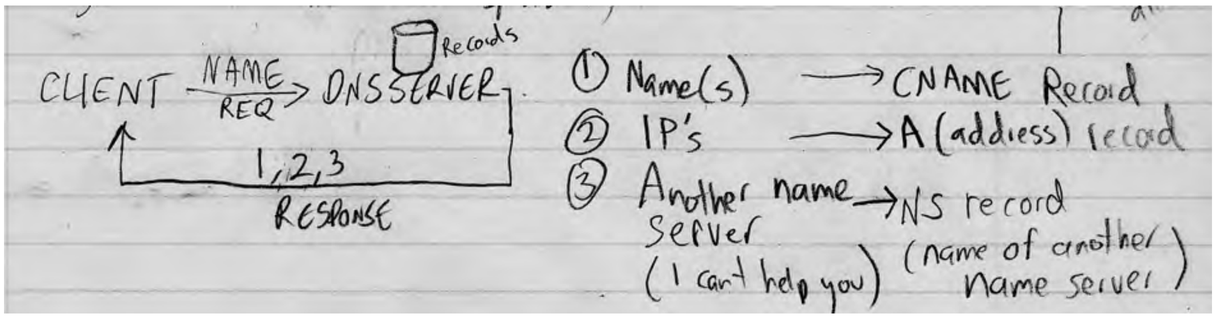

Records
CLIENT
NAME
REQ
DNS SERVER
1,2,3
RESPONSE
① Name(s) → CNAME Record
② IP's → A (address) record
③ Another name server (I can't help you) → NS record (name of another name server)

Figure 6: The Three Possible DNS Entries

This recursive string of calls eventually bottoms out at an IP address. Eventually CNAMEs and NSs will point to a name that resolves to an address.

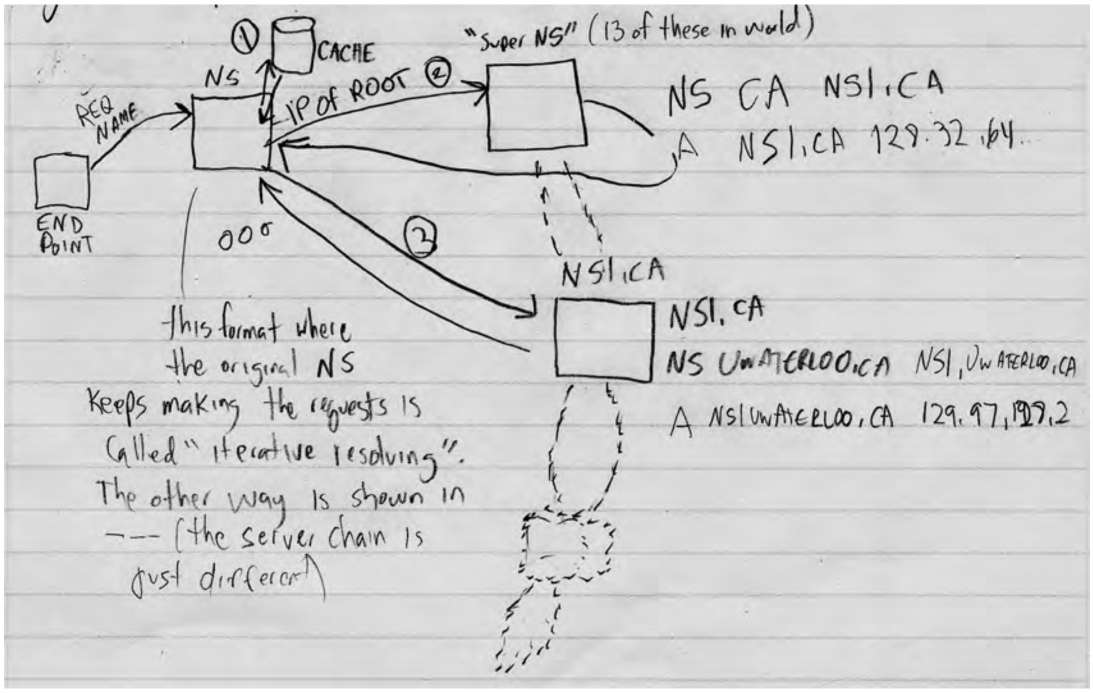
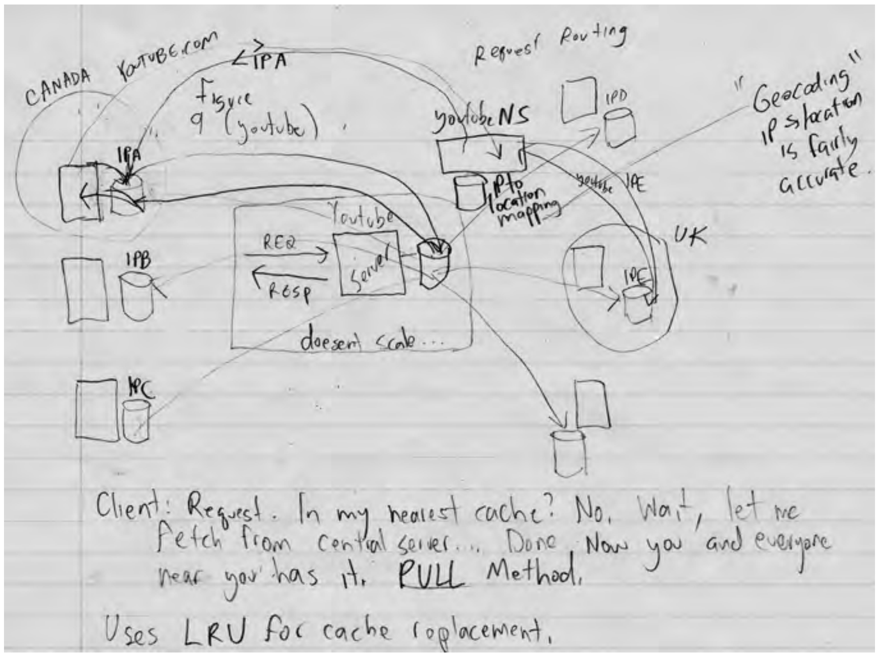

① CACHE
NS
REQ NAME
IP of ROOT ②
"Super NS" (13 of these in world)
NS CA NS1.CA
A NS1.CA 129.32.64..
END POINT
000
③
NS1.CA
NS1.CA
NS UWATERLOO.CA NS1.UWATERLOO.CA
A NS1.UWATERLOO.CA 129.97.129.2
this format where the original NS keeps making the requests is called "iterative resolving". The other way is shown in --- (the server chain is just different)

Figure 7: An Example DNS Trace

2.6 Content Distribution Networks (CDN)

CDNs are networks designed to efficiently distribute content. They follow two main architectures as follows.

2.6.1 Proprietary, like youtube



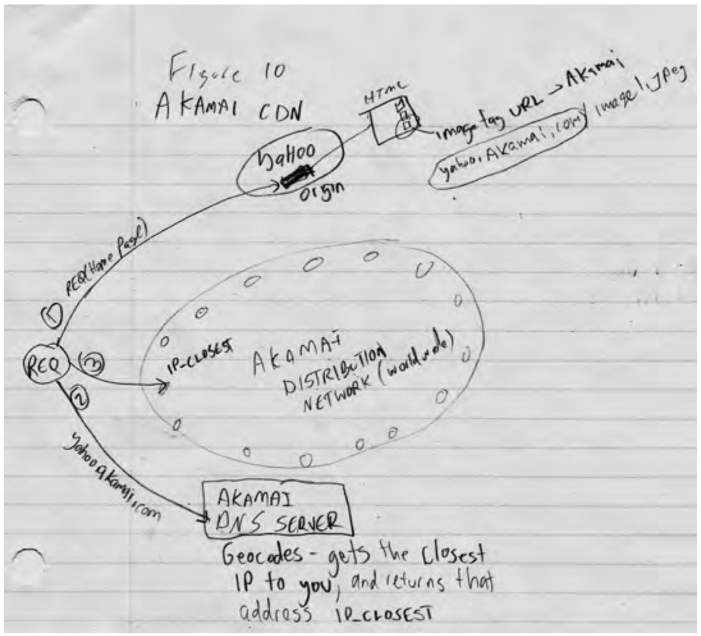
Client: Request. In my nearest cache? No. Wait, let me fetch from central server... Done. Now you and everyone near you has it. PULL Method.

Uses LRU for cache replacement.

Figure 8: A View Of A CDN Such As Youtube

2.6.2 Commercial Rentable CDNs like Akamai

Very Hard to DDOS



Geocodes - gets the closest IP to you, and returns that address IP_CLOSEST

Figure 9: A View Of Akamais CDN

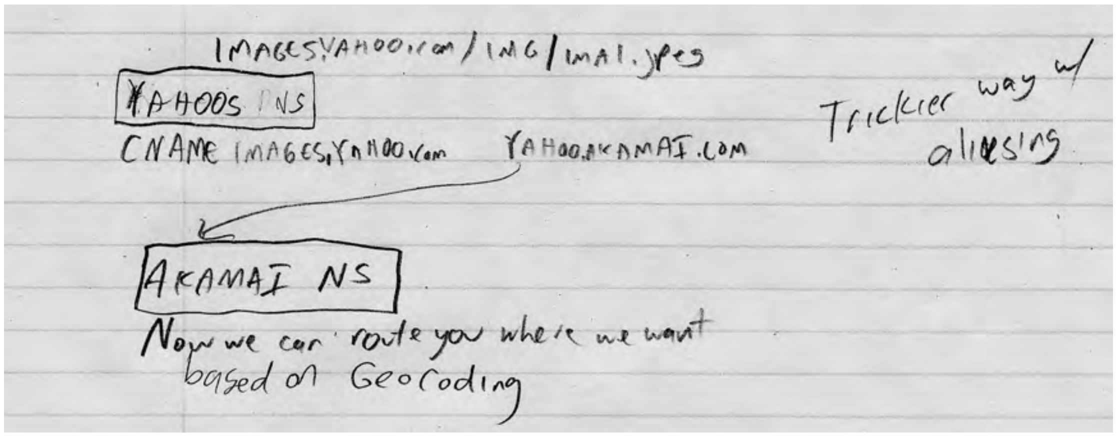

IMAGES.YAHOO.com/IMG/IMA1.jpeg
YAHOOS DNS
CNAME IMAGES.YAHOO.com YAHOO.AKAMAI.COM
AKAMAI NS
Now we can route you where we want based on GeoCoding
Trickier way w/ aliasing

Figure 10: A Tricky Way To Alias Using DNS

2.6.3 P2P

- Goal: efficient and scalable use of resources. Commonly used to share a large number of files.
- Logical graph of connected peers is an *overlay network*.
- Can be many different overlay networks between peers; peers in each of these various overlays have different views of the network
- Multiple copies of content spread out among many users
- Anonymity (not entirely but somewhat)
- Efficient because everyones resources are being used

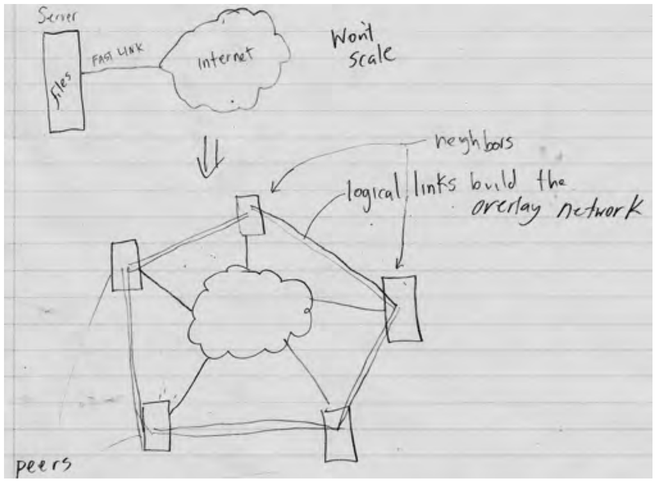


Figure 11: P2P Architecture

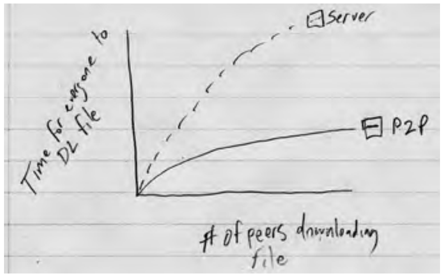


Figure 12: P2P is Efficient At Delivering Content To Many Sources

Bittorrent

- Each torrent has its own logical overlay network
- *Torrent*: key to the overlay network. Contains description of the files that are "in the torrent" and will be downloaded by your Bittorrent client.
- *Tracker*: centralized node (IP is always known) that keeps track of peers in overlay network. Keeps IP addresses, sharing statistics between peers. Can be used to give more bandwidth to peers that are sharing more. Contains IP of trackers that are tracking this torrent.
- *Seeder*: peers that have the entire file and are only uploading the file, whereas *peers* have some of the file but still need some. *Leechers* only download files without sharing.
- μtorrent, VUZE are bit torrent clients that implement the Bittorrent protocol.
- Files are split into chunks, typically of size 256kb
- Peers in the same torrent overlay network share info about who has what chunks every 10 seconds
- Rarest chunks are downloaded first to populate the network with the file and reduce bottlenecks
- Leechers are eventually dropped; Bittorrent clients are built to stop communicating with peers they are not receiving anything from
- Freeriding: once a leecher has been dropped from a bunch of peers, they can go request even more peers from a tracker and grab the remaining chunks they still need from that new set of peers.

Seaching within P2P networks

- Centralized Method:
  - Can have a centralized directory service that keeps track of all files being shared and who has them (Napster).
  - When a peer wants a file, they ask the DS who has the file and obtains the IPs
  - Advantages:
    - You don't need any search engines; the DS has all the info you need
    - Ease of programming: very simple model and thus easier to implement
  - Disadvantages:
    - One point of failure. The police find that one server and everyones [explative]
    - Very hard to scale. Must replicate DS and so on
- Totally Decentralized:
  - Requests are flooded throughout network. Peers with the requested file contact the original requester directly.

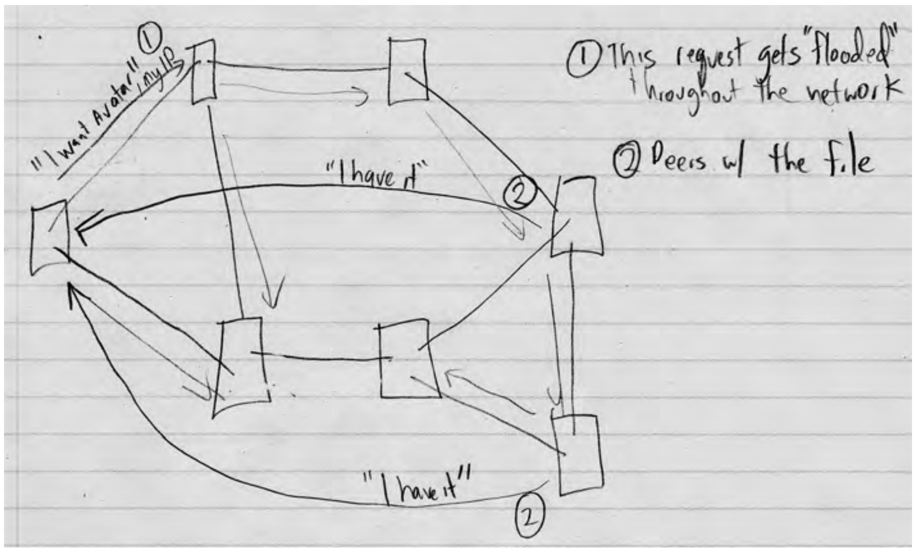


Figure 13: A Totally Decentralized Approach To P2P Searching

- Bittorrent: some combination of the two

Uses

- Ilicit file sharing...
- Skype
- Can be used internally within data centers

2.7 Datacenters / "The Cloud"

2.7.1 What is the cloud?

- A datacenter
- Centralized content
- Online apps
- Remote computing
- Renting compute power (EC2)

2.7.2 What is a datacenter (physical view of cloud)?

- Warehouse full of racks full of servers
- Largest datacenter today (Microsoft's Chicago DCN): 200,000 servers
- Cooling has been the major cost. Now people are moving them to freezing countries and just leaving the window open!

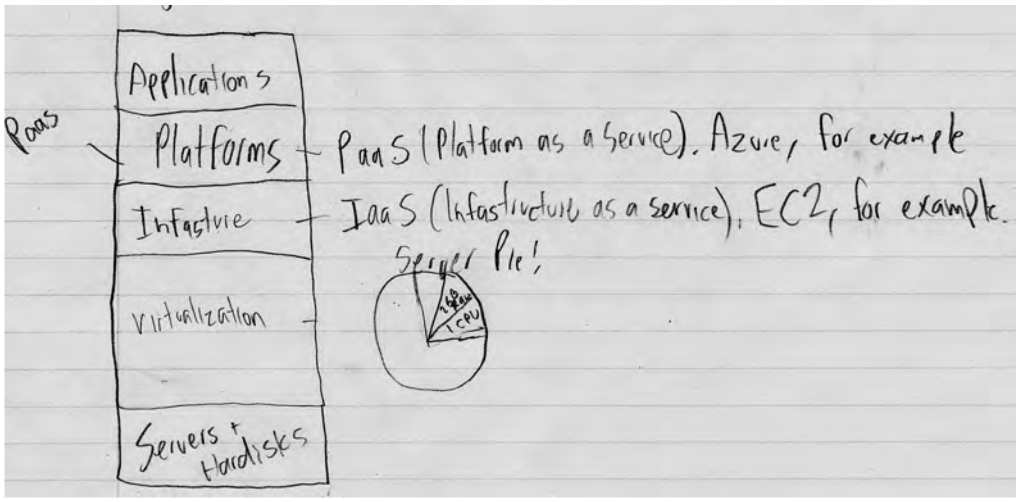

Applications
Platforms
Infastrure
Virtualization
Servers + Hardisks
PaaS
PaaS (Platform as a Service). Azure, for example
IaaS (Infastructure as a service). EC2, for example.
Server Pie!
1 CPU

Figure 14: Logical View of DCN Layers

2.7.3 Service View (Cloud Operations)

- Gmail, iCloud, hotmail, Amazon Wispernet, dropbox, Siri, youtube (uses DCs to distribute content)

2.7.4 PaaS: Platform as a service

- Windows Azure for example. Offers things like auto scaling, reliability, redundancy for your website

2.7.5 IaaS: Infastructure as a service (renting computing power)

- EC2 Web Services, for example. You rent a (or part of a) server. Disadvantage: does not offer autoscaling and other things platforms offer. However, you have much more freedom to do whatever you want with the server.

2.7.6 Why is cloud computing "hot"?

- Virtualization allows you to slice up a server into multiple pieces that multiple people can use. Allows for very efficient use of resources.
- Remote computing: users can access their apps, data, and can utilize DCN hardware from anywhere. I can run some crazy hard algorithm in the cloud from my mobile phone.
- Stability, redundancy.
- Scaleable to load: as demand for an app increases, you can dynamically use more computing power
- Cost: renting is less expensive than buying your own server, cooling, maintenance etc.

2.8 Socket Programming

- http://en.wikipedia.org/wiki/Network_socket
- app→API→pipe→API→app
- This API, known as sockets, is provided by the OS
- Server sets up a "window front" that is open to requests and listens. Clients come to the store (send a request) when they wish. Asynchronous relationship.
- A socket is essentially a handle on the pipe, like a file handle. R/Writing from the socket is like R/Writing down the pipe.

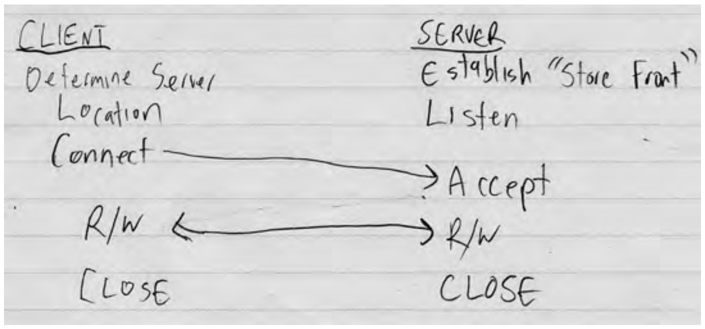


Figure 15: Socket Programming

2.8.1 The Socket Structure for Keshav's C code

- SOCKET: Just an index into the SOCKET TABLE. It is essentially just an int that points to the corresponding row in the table.
- SOCKET TABLE: Keeps information about currently running sockets.
- SOCK ADDRIN: *The entries stored in the socket table.* Three subfields:
  - ADDRESS FAMILY: AFINET (fixed)
  - IP ADDRESS: "Any". States who I am accepting connections from.
  - PORT NUMBER: Servers port - "listen on this port".
- HOST ENT: What is returned by DNS. Part of this is the IP and will be copied into the above table

3 Transport Layer

3.1 Main Purpose

Logical communication between applications. To apps running on various networked devices, it is as if they are directly connected to each other. It doesn't matter if the devices are 10 feet or 6,000km from each other.

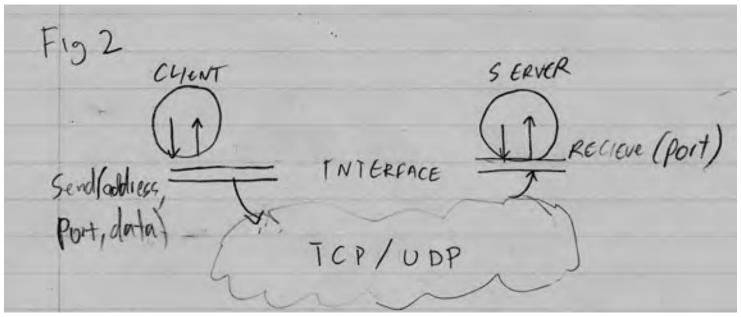


Figure 16: The Magical Transport Layer Pipe Between Apps

3.2 Definitions

- *Headers*: Information containing metadata about packet. Indicates destination, source, among other things.
- *Segments*: Transport layer piece of the message. Known as a datagram in the network layer. We break apart messages because we have a connectionless pipe. If we try to send one massive message and there is congestion, there is a chance we will lose our entire packet.
- *Buffers* along the path between SRC and DST queue packets until they can service them.
- *Ack*: acknowledgement that a packet was received by DST
- *Sequence Numbers (TCP)*: Assigned for each packet for re-assembly by DST
- *Round Trip Time (RTT)*: time for a packet to be sent and acknowledged.
- *Timeout (TCP)*: the amount of time SRC waits for an Ack before it sends its packet again. Timeout > RTT obviously.

3.3 Services that *can* be offered by Transport Layer

- *Reliable delivery*: can ensure packets lost in the network layer are re-transmitted.
- *Flow control*: control the amount of data sent between two nodes so one does not overwhelm the other.
- *Congestion control*: control the amount of data sent in a network to ensure it is still functional

3.4 Multiplexing and Demultiplexing

Multiplexor: Many incoming signals → MUX → one output channel. Combines many incoming signals onto one channel via various mechanisms (time delayed for example).

Demultiplexor: One incoming channel with many signals on it → DEMUX → many different channels. Separates many signals coming in on one channel onto separate channels.

3.5 Port Multiplexing

The transport layer multiplexes the many signals coming from applications using *port numbers*. Ports:

- Are a unique identifier of applications on an end-host
- 2^{16} different port numbers
- 0-1023 are reserved for well-known services. Forex 80=http, 21=ftp

3.6 UDP

User datagram protocol. When it was originally named datagrams was the name used for segment in transport layer, but then they were later renamed to segments...

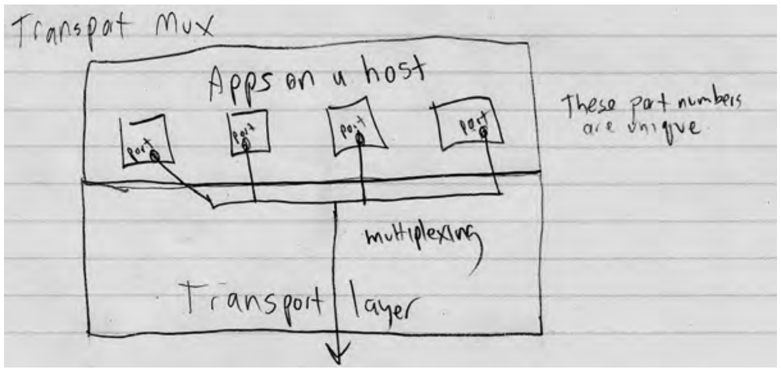


Figure 17: Transport Layer Port Multiplexing

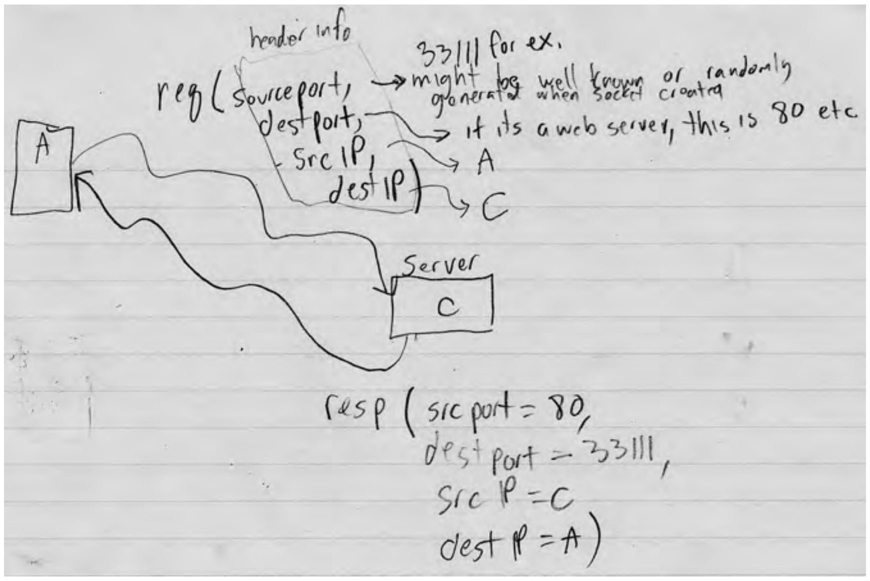


Figure 18: Identifying Applications Via Ports

3.6.1 UDP Design Goals

- Designed to be as simple as possible.
- Connectionless protocol. As soon as there is data to send from an app, it blasts it down the pipe. Does not handshake with the end host
- Does not provide reliability, flow control, or congestion control
- Does not guarantee in order delivery.

3.6.2 UDP Header (8 bytes)

1. Source Port (16 bits). Actually this is optional.
2. Destination Port (16 bits). Required.
3. Length (16 bits). Length of payload.
4. Checksum (16 bits). Optional for IPV4, Required for IPV6. Primitive error detection. If some number of bits was corrupted, then UDP just drops the packet. "Sorry, I Tried"

3.6.3 Actual Uses for UDP

- It is faster because you have fine grained control. You can write to the network as fast as you want to and packets are sent immediately.
- No handshake (even faster).
- No connection state: much less overhead.
- Small packet overhead (header is only 8 bytes). Less bits need to be sent. TCP has at least 2.5x the header. Good for applications that send a lot of small updates that do not matter if one is lost
- Good for real-time applications. Video games tend to use UDP.

3.7 TCP (1974)

Won a Turing Award.

TCP Provides reliability (Section 3.8), flow control (Section 3.9), congestion control (Section 3.10), and in order delivery. Downside is that it is quite complicated. In this section we present some various details, but those sections have the majority of TCPs contents.

3.7.1 Closing

Two ways to end a TCP connection.

1. SRC sends FIN, DST ACKS FIN, SRC ACKS FIN. Done
2. timeout

3.7.2 Preformance Results Example

Throughput = transmission Rate. Throughput =

$$\frac{w}{RTT}$$

Let $W = w$ when packet loss occurs. Assume loss is always detected by triple ACK (see Congestion Control for details). Also assume RTT and W are constant for duration of connection.

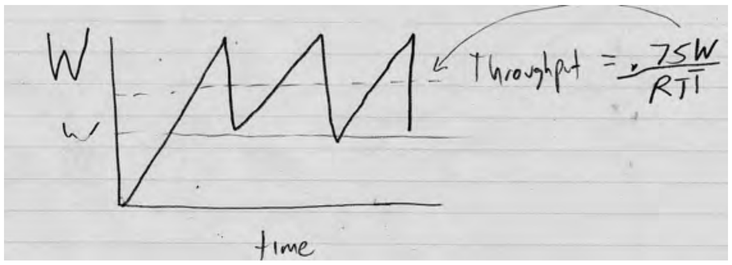


Figure 19: Simple Performance Example

3.7.3 Drawbacks of TCP

TCP has some drawbacks because the only info it has to make its decisions (on flow, congestion control etc.) is packet loss. We *have* to overflow buffers to find the amounts of available bandwidth. The way around this would be to use explicit congestion notification, further discussed in Section 3.10. The problem with this is that it requires all devices in the network implement this protocol.

- Fairness can be gamed. Malicious users can steal bandwidth.
- TCP assumes packet loss means congestion. This is not true, forex in wireless links, packets may be lost just due to interference. Also, this problem can be prevalent in high speed links (suppose 10Gbps). Suppose each packet is 1500 bytes. To maintain a 10Gbps sending rate, we need a massive window size (w = 83,333). Whenever we have a packet loss we cut our window in half though, which means to maintain this rate we need to have a packet loss only every 5Billion packets!
- TCP detects congestion using loss. TCP has to cause packets to be dropped to determine when the network is congested. The buffer at the bottleneck will nearly always be full because of this.
- Sensitive to the threshold at which we switch from exponentially increasing the window size to an additive increase.
- Starting with a window size of 1 is outdated. Google proposes starting the initial window size at 10 packets, or roughly 10.5kb. In this case, you might only need one round trip to fetch a webpage, vs. the three you would need if you were using slow start starting at a window size of 1.
- Open to SYN flooding DoS attack.

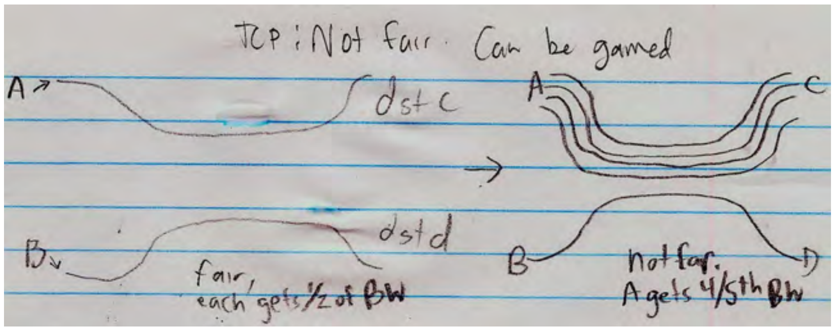


Figure 20: Opening Multiple TCP Connections Can Game Bandwidth

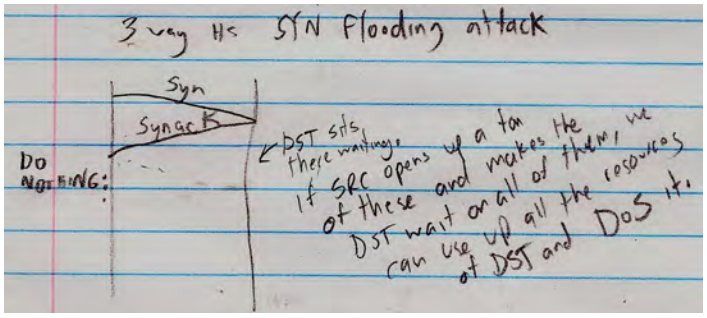

3 way HS SYN flooding attack
Syn
Synack
DO NOTHING:
DST sits there waiting.
If SRC opens up a ton of these and makes the DST wait on all of them, we can use up all the resources of DST and DoS it.

Figure 21: TCP Syn Ack Attack

3.8 Reliability

Reliability - we want to be sure messages sent are delivered. The problem is that buffers along various paths in a network might be full, and when this happens, packets are dropped.

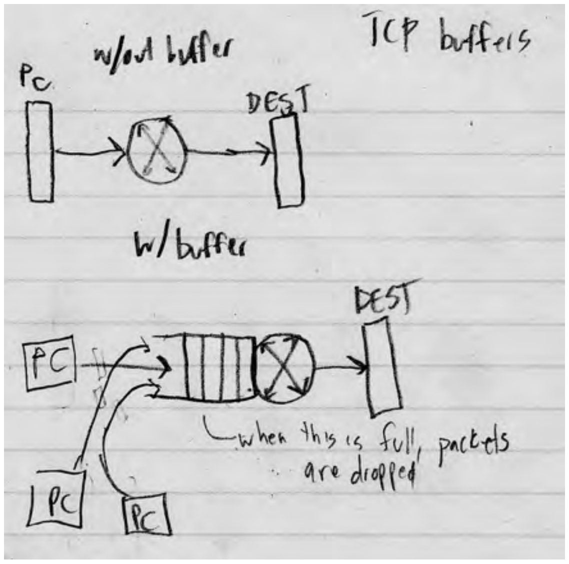


Figure 22: Packets Are Dropped When Buffers Are Full

Reliability via Multiple Copies: Can simply send more copies of a packet. $p(delivery) = errorrate^{numcopies}$. Problem: this probability still never reaches 1, and you need to send a lot more data!

Reliability via Acknowledgements:
"Stop and wait":

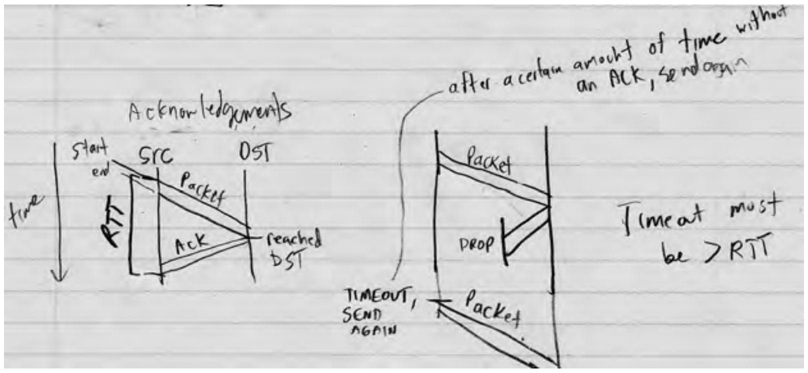


Figure 23: TCP Acknowledgements Idea

Problem: we will waste time waiting around for ACKs if we send one at a time. Better to send a bunch and wait for one ack.

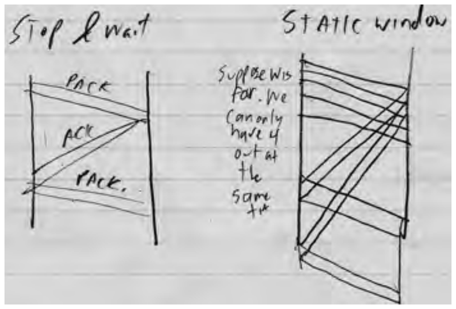


Figure 24: Stop & Wait vs. Window

Reliability via Cumulative Acks w/ Sequence Numbers:
First, we set up a connection via a "3-way handshake".

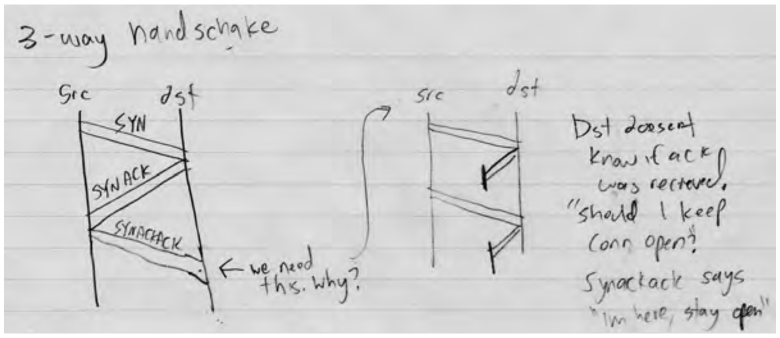

3-way handschake
Src
dst
SYN
SYN ACK
SYNACKACK
← we need this. Why?
src
dst
Dst doesn't know if ack was received, "should I keep conn open?"
Synackack says "I'm here, stay open"

Figure 25: 3-Way Handshake

Then we can use cumulative acks and sequence numbers to ensure reliability. The idea is to send many packets at once, and let the DST acknowledge the last in-order packet it received. Sender will then send another batch starting at that ack number.

3.9 Flow Control

Techniques to match a SRC sending rate to the DST receiving rate.
We want a mechanism that is

- Simple
- Efficient (minimal overheads)
- Fair: forex, 4 senders and one receiver, each sender should send at 1/4th of the receivers rate.
- Stable: converge to equilibrium.

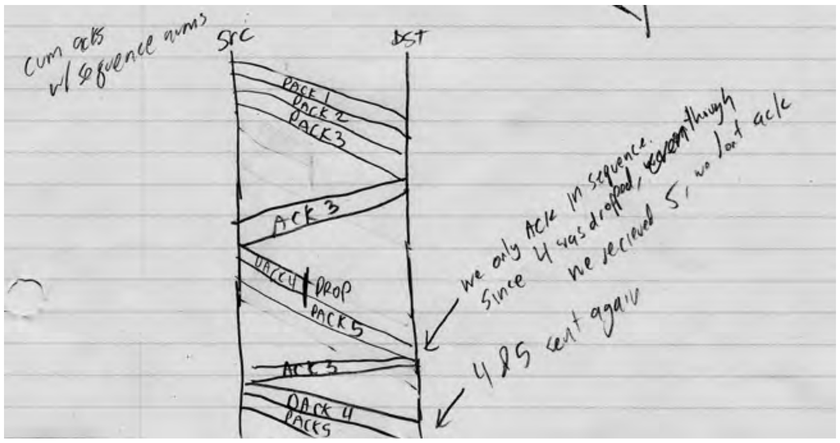

cum acks w/ sequence nums
SRC
DST
PACK 1
PACK 2
PACK 3
ACK 3
PACK 4
DROP
PACK 5
ACK 3
PACK 4
PACK 5
we only ACK in sequence. Since 4 was dropped, even though we recieved 5, we don't ack
4 & 5 sent again

Figure 26: Cumulative Acknowledgements Idea

3.9.1 Open Loop Flow Conrol

No feedback between SRC and DST. This can be accomplished using *admission control*: only allow SRCs in that you can accommodate. SRCs ask for requirements such as bandwidth, latency, and the DST admits / rejects requests based on what they can accommodate. The SRCs that were accepted then send at the agreed rate until it is done.

Admission control has several problems. It was used for dialup, a technology we can hope to soon forget ever existed for the sake of civilization.

3.9.2 Closed Loop Flow Control

Closed Loop: sender and receiver dynamically settle to a sending rate. Devices do not reserve bandwidth ahead of time. Instead we use *statistical multiplexing*.

How do we do this? Approaches:

1. Network devices use *explicit control messages* to negotiate sending rate. These control messages are actually put in the header of packets. There is a field that says whether there is congestion in the network.

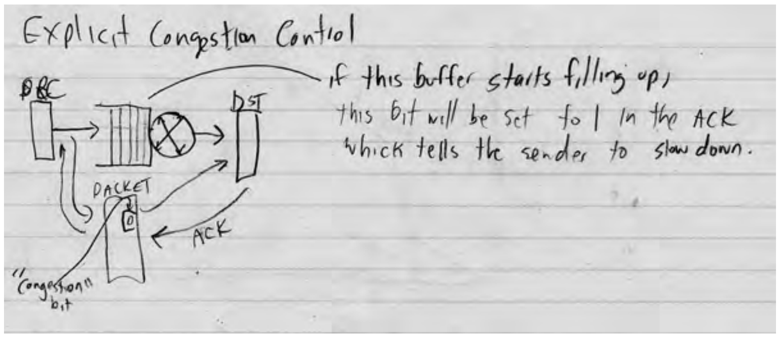

Explicit Congestion Control
SRC
DST
if this buffer starts filling up, this bit will be set to 1 in the ACK which tells the sender to slow down.
PACKET
0
ACK
"Congestion" bit

Figure 27: Explicit Flow Control

2. *Implicit flow control.* Dynamically adjust transmission window in response to undelivered packets. Discussed further in TCP Congestion Control.

3.10 Congestion Control

In data networking and queueing theory, network congestion occurs when a link or node is carrying so much data that its quality of service deteriorates. Eventually we can reach Congestive collapse (or congestion collapse): a state of a packet switched computer network where little or no useful communication is happening due to congestion. Congestion control concerns controlling traffic entry into the network, so as to avoid congestive collapse. It does so by attempting to avoid oversubscription of any of the processing or link capabilities of the intermediate nodes and networks and taking resource reducing steps, such as reducing the rate of sending packets. It should not be confused with flow control, which prevents the sender from overwhelming the receiver.: http://en.wikipedia.org/wiki/Network_congestion_avoidance

3.10.1 Flow Control Vs Congestion Control

Flow control means preventing SRC from sending data that the DST will drop because it runs out of buffer space. This can be done with a sliding window protocol–we just make sure the SRCs window is no larger than the free space in the DST buffer. TCP does this by letting the sink advertise its free buffer space in the window field of the acks.

Congestion control means preventing (or trying to prevent) SRCs from sending data that will end up getting dropped by some intermediate buffer or router in a network because its queue is full. This is more complicated, because packets from different sources traveling on different paths can all be filling into the same queue.

3.10.2 TCP Congestion Control

Transmission Control Protocol (TCP) uses a network congestion avoidance algorithm that includes various aspects of an additive increase/multiplicative decrease (AIMD) scheme, with other schemes such as slow-start in order to achieve congestion avoidance. from `http://en.wikipedia.org/wiki/TCP_congestion_avoidance_algorithm`

I don't feel like typing everything about TCPs congestion control algorithm because it would be like re-inventing the wheel. There are several PPTs freely available. Here are some good ones.

- Heres a really good one: `http://inst.eecs.berkeley.edu/~ee122/fa06/notes/18-Congestion-6up.pdf`
- `www.cs.uiuc.edu/class/fa06/cs438/slides/CS438-18.CC.ppt`
- `www.ipam.ucla.edu/publications/cntut/cntut_1497.ppt`
- Here is the actual RFC: `http://opalsoft.net/qos/TCP-1010.htm`

3.11 Apps and their Transport Layer Protocols

app name(app layer protocol): transport layer protocol

- Email (SMTP): TCP
- Web (HTTP): TCP
- File Transfer (FTP): TCP
- Remote File Servers (NFS): typically UDP
- Streaming (generally proprietary): UDP
- Voip, Skype (typically proprietary): UDP
- Network Management (SNMP): UDP. In this case we don't actually wan't TCP throttling us because this application needs to be run when there are problems in the network

3.12 What Is An Ideal Transport Protocol?

Depends on the network!

- Suppose we want to communicate with a rover on the moon. The RTT is 2.5seconds at least. Suppose we have a 100Mbps link. 100mb$\times$2.5sec = 256Mb = 32MB. We could send 32MB before we receive any acknowledgements. An ideal protocol would take advantage of this fact.
- In DCNs, we have high bandwidth and a low RTT. Microsoft: use TCP+Explicit congest. notifications (ECN). Because DCNs are owned by one party, they can buy switches that use ECN (because they own all the servers, they can enforce all of them support ECN. Now we have 3 congestion signals: timeout, triple dup ack, ECN.
- Consider wireless networks. Wireless networks have very high loss rates just due to the channel quality itself. TCP is not ideal in this situation because it considers lost packets as a sign of congestion. We can use the link layer to help TCP in this scenario: we can let the link layer quickly retransmit lost segments.
- TCP works well for many cases, forex wide area networks (WAN). It has been tested extensively and we know its limitations. After 30 years of research, we haven't found a better protocol!

4 Network Layer

4.1 Main Purpose

The network layer is responsible for routing datagrams within networks. Datagrams are like postcards. Everything you need to know about how to deliver a datagram is contained in the datagram.
We learned to uniquely identify any app in a network, we need a port number and an IP address. We saw the transport layer is responsible for dealing with port multiplexing. It is responsible for sending packets to the right apps once they arrive at a host. The network layer is responsible for routing packets through the network to those hosts, and deals with IP addresses.

4.2 IP Addresses

The narrow part of the hourglass model (everything goes through IP)

- IPV4 = 32 bits in the form of a.b.c.d, a,b,c,d $\in [0, 255]$
- Public: reachable from everywhere
- Private: unreachable local addresses that have no meaning outside of your domain. These are enforced to be non-routable on the internet

4.2.1 The IP Header

See `http://www.networksorcery.com/enp/protocol/ip.htm` for the exact IP header information. Here are some notes that are not in there:

- Keshav: "The TOS (renamed to Differentiated Services in IPV4) field was a miserable failure and sunk billions of dollars and thousands of hours of research time".
- When the TTL field hits 0 (it is decremented by 1 at every router), an ICMP (internet control message protocol) message is sent to the source of the packet informing them"your packet expired".
- MF attack. Send large packets, but never set MF to 0. This makes the receiver wait and wait for my monster packet to be completed, but it never happens. This vulnerability has now been fixed.
- Header Checksum info: `http://en.wikipedia.org/wiki/Header_checksum`

4.2.2 Multiple IPs

Anything with two interfaces can become a router. Routers simply have two IPs: an "incoming" and "outgoing" IP. Suppose your iPhone has a 3G and a wifi connection: then it has 2 IPs. You could write software that takes packets destined to one of those IPs and forwards it out onto the other.

4.2.3 How To Get An IP

- Allocation: your ISP (like Rogers) gets a block of IPs from IANA (Internet Assigned Numbers Authority). They control the space of all IP addresses and have smaller registrars that deal w/ regional chunks (forex ARIN controls N.A.) whois keeps track of these organizations like ARIN.
- Hand them out: DHCP. A DHCP server on a subnet knows about the current available pool of IPs within that subnet. When end hosts in the network need and IP, they broadcast that message, "I need an IP". The DHCP server then assigns you an IP. This also works on the higher tier: Rogers has a DHCP server that hands your home gateway and IP address. Your home router is handing out IPs like 192.168.1.* to all of your internal NATted devices, whereas rogers is giving you your home IP like 128.x.x.x.

4.2.4 Network Masks

- Used to refer to multiple addresses in a single line. For example. 129.97.1.* refers to all 256 IPs $\in \{192.97.1.0, ..., 192.97.1.255\}$.
- The reason we use a mask like 129.97/16 instead of 129.97.$*$ in routing tables is because its hard to represent $*$ in binary.... So instead we know how many addresses we have from the number after the slash.
- Suppose my address is a.b/16. This means I have 2^{32-16} routable addresses in my network.

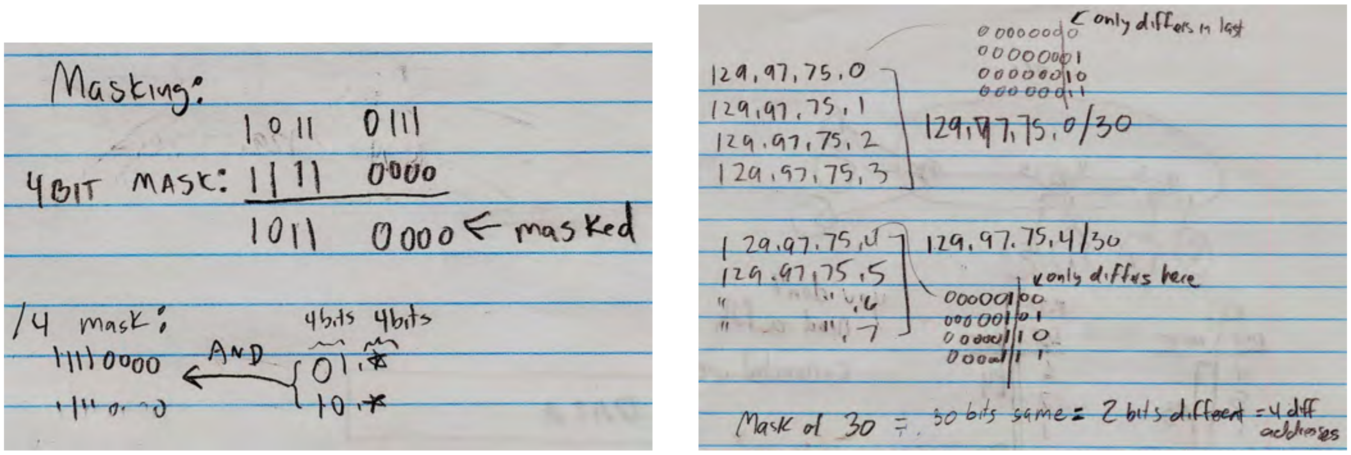


Figure 28: Left: Bit Mask. Right: A /30 Mask Example

4.2.5 Geographical Location of IP Addresses

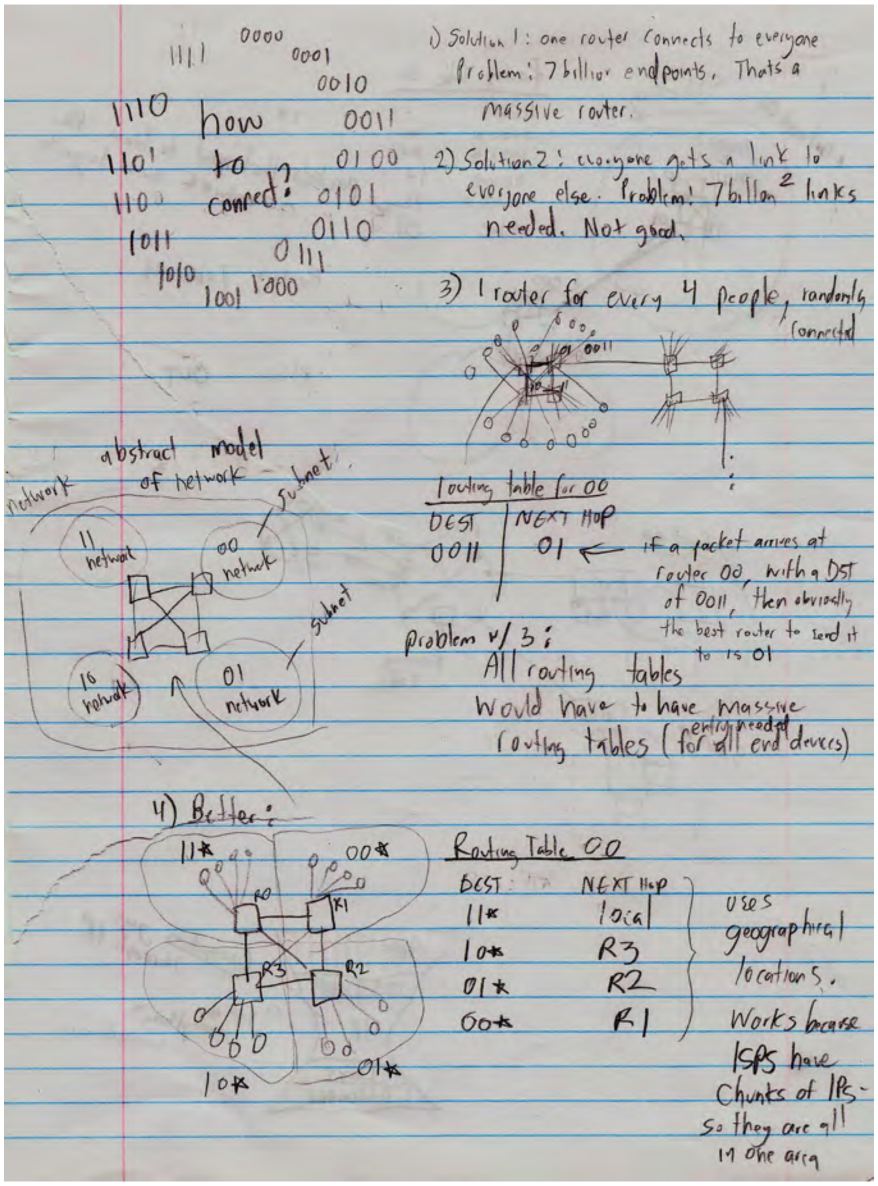

how to connect?
1111 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110
1) Solution 1: one router connects to everyone. Problem: 7 billion endpoints. That's a massive router.
2) Solution 2: everyone gets a link to everyone else. Problem: 7 billion² links needed. Not good.
3) 1 router for every 4 people, randomly connected
abstract model of network
network
subnet
11 network
00 network
10 network
01 network
routing table for 00
DEST | NEXT HOP
0011 | 01
if a packet arrives at router 00, with a DST of 0011, then obviously the best router to send it to is 01
problem w/ 3: All routing tables would have to have massive routing tables (entry needed for all end devices)
4) Better:
11* 00* 10* 01*
R0 R1 R2 R3
Routing Table 00
DEST | NEXT Hop
11* | local
10* | R3
01* | R2
00* | R1
uses geographical locations. Works because ISPs have chunks of IPs - so they are all in one area

Figure 29: Better and Better Ways to Organize a Network - Geographical Location is Best

4.2.6 IP Spoofing

The SRC field on the IP packet is optional. If you don't want a response, you don't need to put one. If you want a response, you obviously need it. IP Spoofing means changing the SRC. For example, an attack is to send millions of requests asking for a response, but give the SRC as someone else. This will direct millions of packets to that request.

4.3 Subnetting / Heirarchical Networks

4.3.1 Three Level Hierarchy

The internet today uses a three tier hierarchy. See that we have the *core*, *networks*, and *subnetworks*

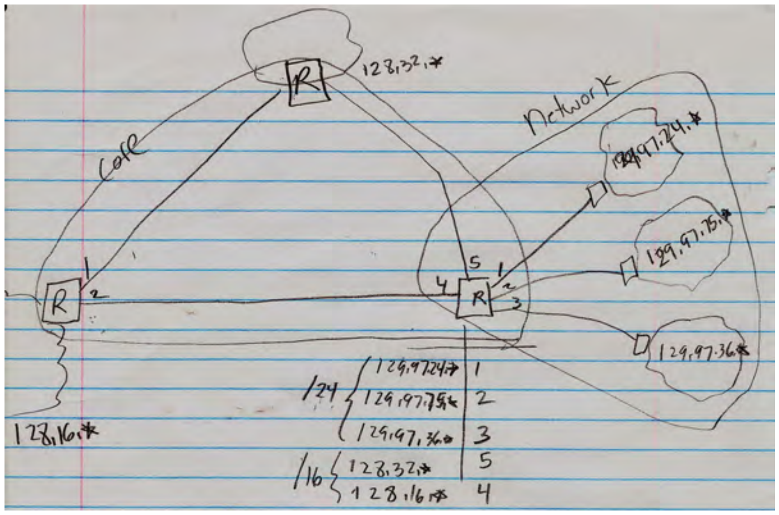


Figure 30: Three Tier Hierarchy

4.3.2 Subnetting

The act of splitting one massive network into logical divisions that make routing easier is known as subnetting. Subnetting refers to grouping IP addresses by location, and then using masking to refer to groups as a whole. It drastically reduces the size of routing tables, because routers only need one entry for all packets designed for some subnetwork.

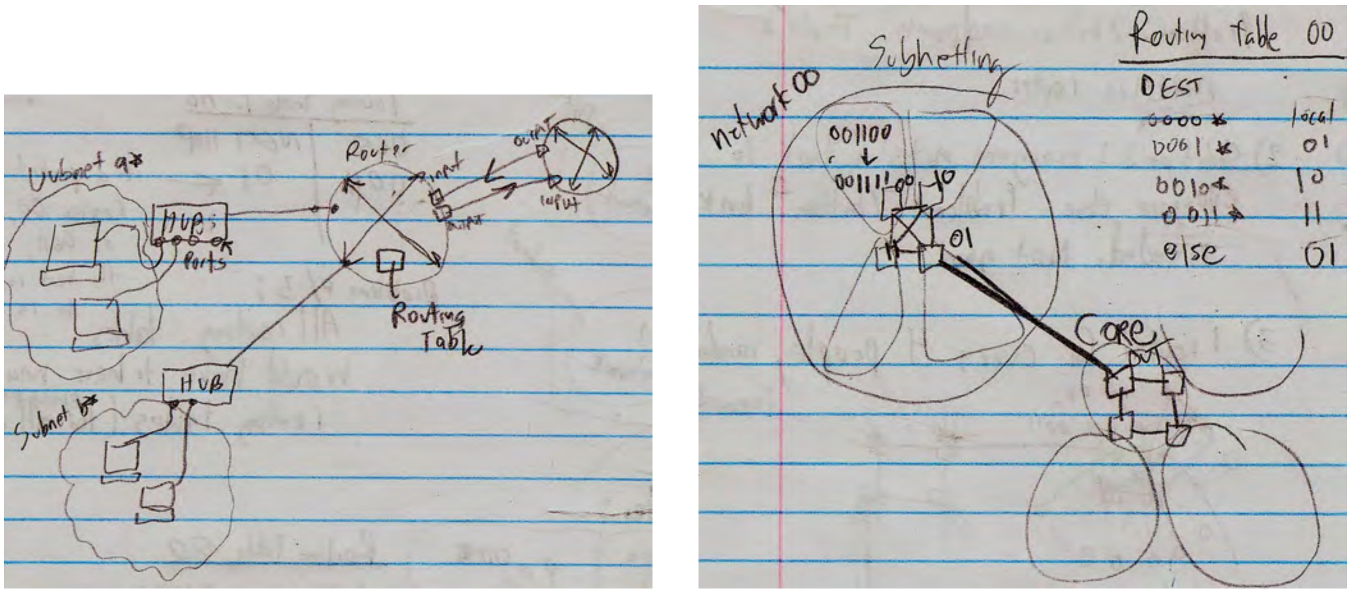


Figure 31: Subnetting Figures 1 & 2

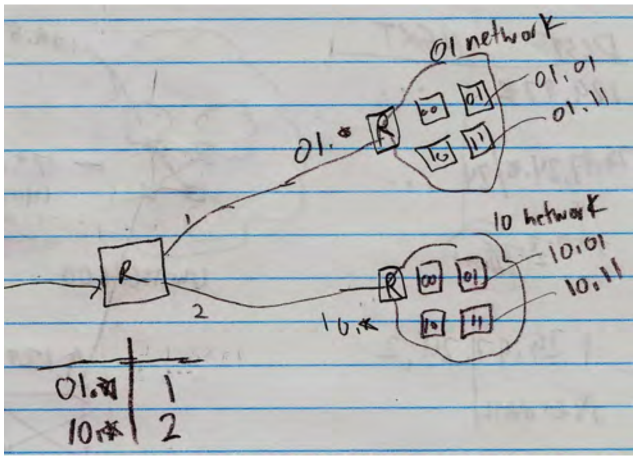


Figure 32: Two Subnetworks

We now combine subnetting with the network masking slash notation (the "*" notation is helpful but not actually used):

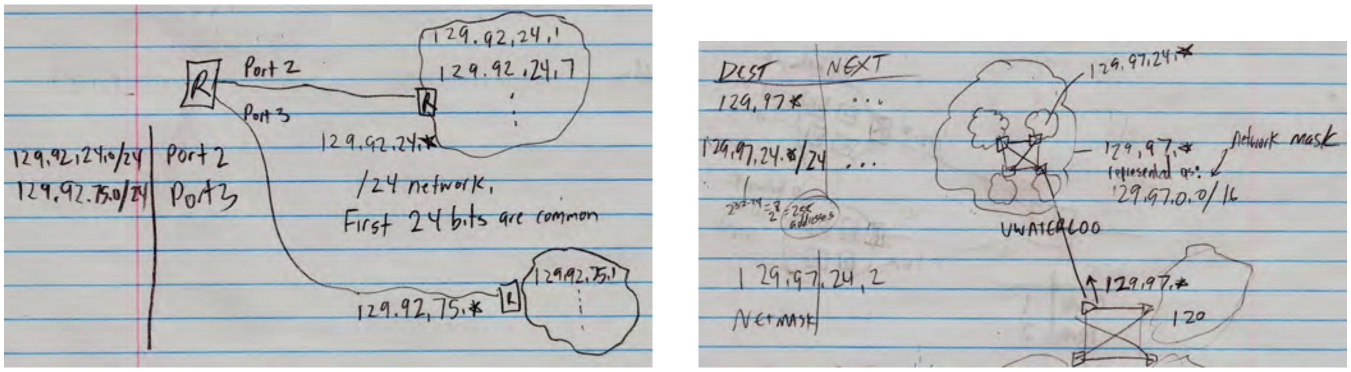


Figure 33: Subnetting Using The Network Mask Slash Notation

4.3.3 The Core

The "core" can be thought of the collection of routers at the "heart" of a network. Didn't talk too much about this yet.

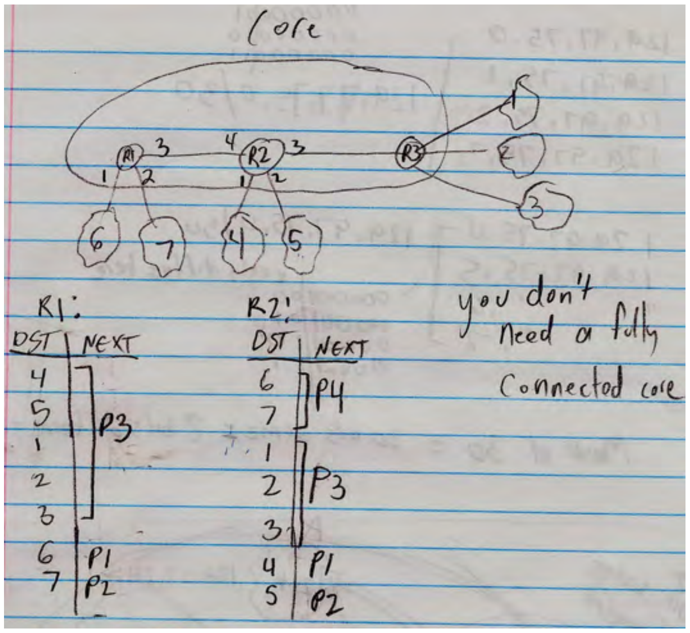


Figure 34: The Core Does Not Need To Be Fully Connected

4.4 Abstraction

Networks are complicated; they are composed of many routers, switches, subnetworks, links etc. In order to understand some network concepts, it is useful to be able to abstract some complexity from network diagrams. Network designers have come up with some abstractions that make the explanation, implementation, and teaching of networks easier.

4.4.1 Areas & Asynchronous Networks

UW's ISP only needs to know about UW's external IP address so that it can send all traffic destined to hosts inside of UW to UW's routers which will then handle the requests. The ISP does not want/need to think of all of the equipment inside of UW; routers, switches, links etc. To the ISP, UWs internal network contents can be abstracted into one "blob" called the UW asynchronous network. We can further apply abstraction within this network by dividing the University into "areas", for example computer science, engineering, etc.

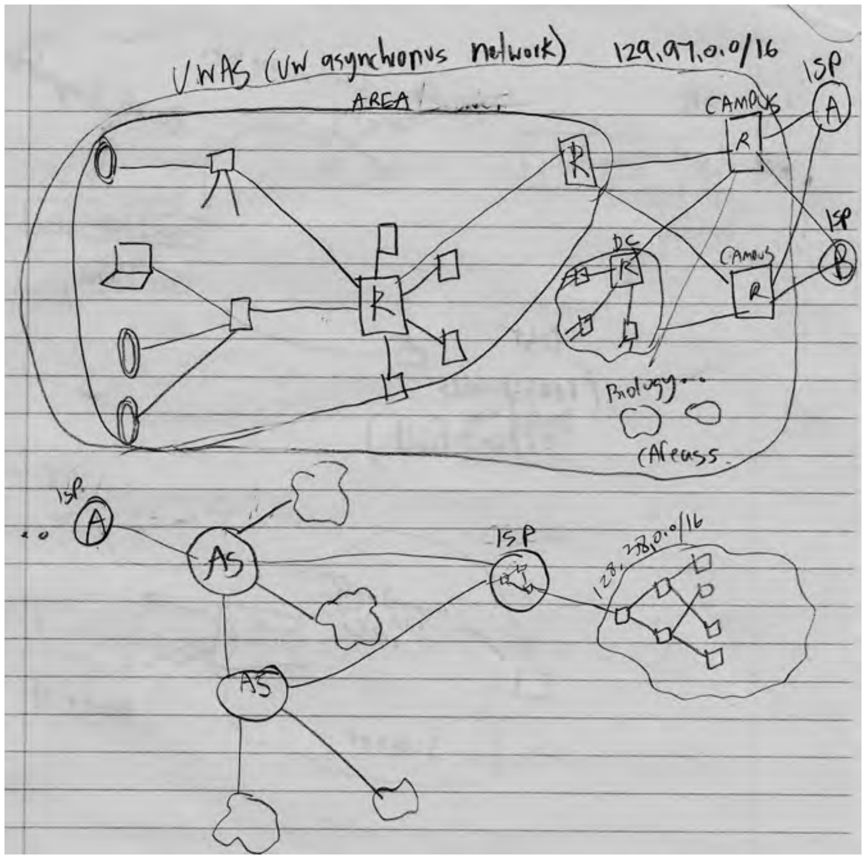


Figure 35: Asynchronous Network and Areas

4.4.2 Graph Abstraction

Many routing principles can be explained using a graph theoretic notation. For example, some routing algorithms are based on the shortest path graph algorithm. Many algorithms are related to partitioning of networks (although we may not discuss them in this class), which are analogous to partitioning graphs. We can abstract networks into a graph theoretic notation to make understanding these algorithms easier, as shown in the following diagram. This graph representation is useful for explaining a variety of networking concepts and algorithms.

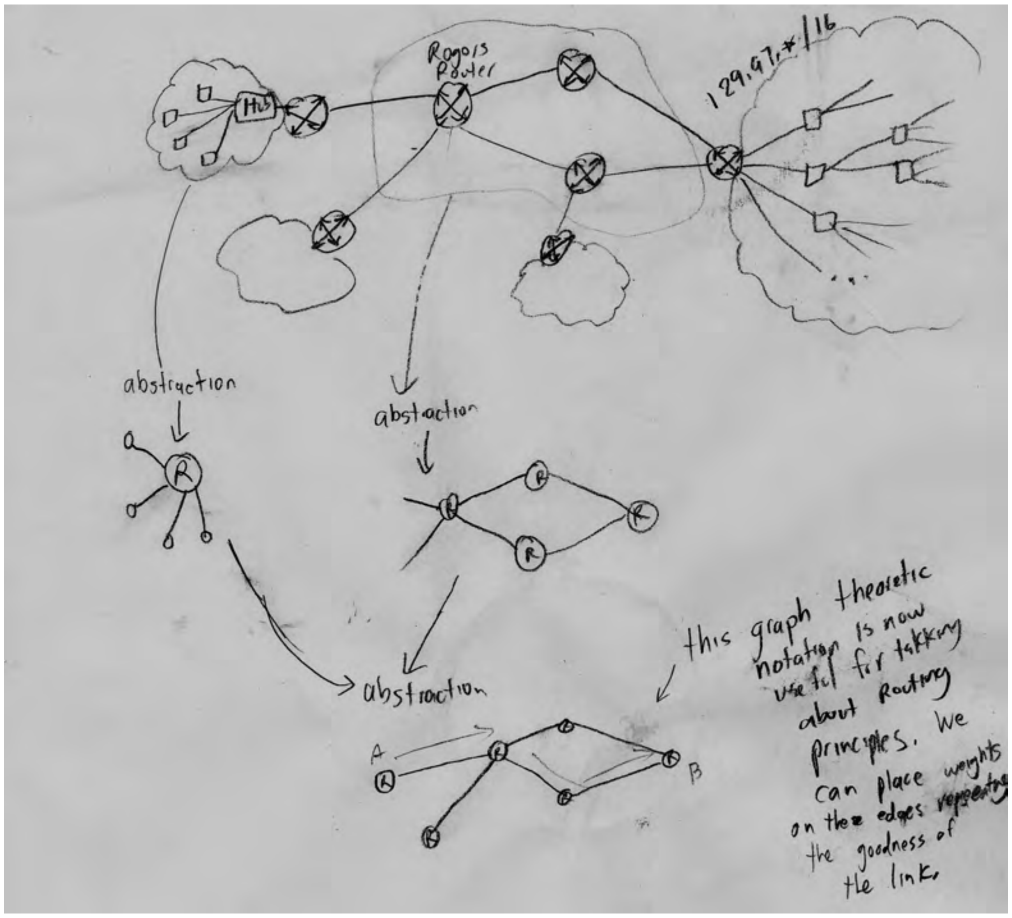

Hub
Rogers Router
129.97.*/16
abstraction
abstraction
abstraction
A
B
this graph theoretic notation is now useful for talking about Routing principles. We can place weights on the edges representing the goodness of the link.

Figure 36: Abstraction from a network into a graph

4.5 Routing Principles

4.5.1 Fragmentation

Sometimes a router will have limits as to the size of the packets that it can send. In the case the router receives packets bigger than it can send, it has to preform packet *fragmentation*; splitting a packet into chunks and then populating certain fields in its header so that the receiver may concatenate the chunks.

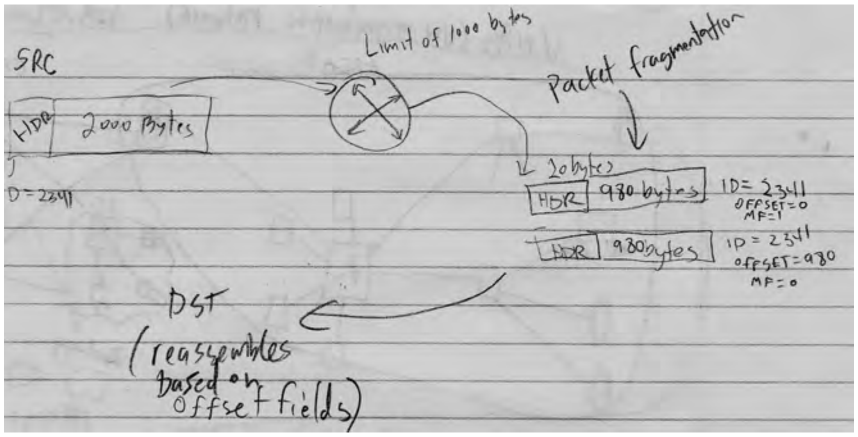
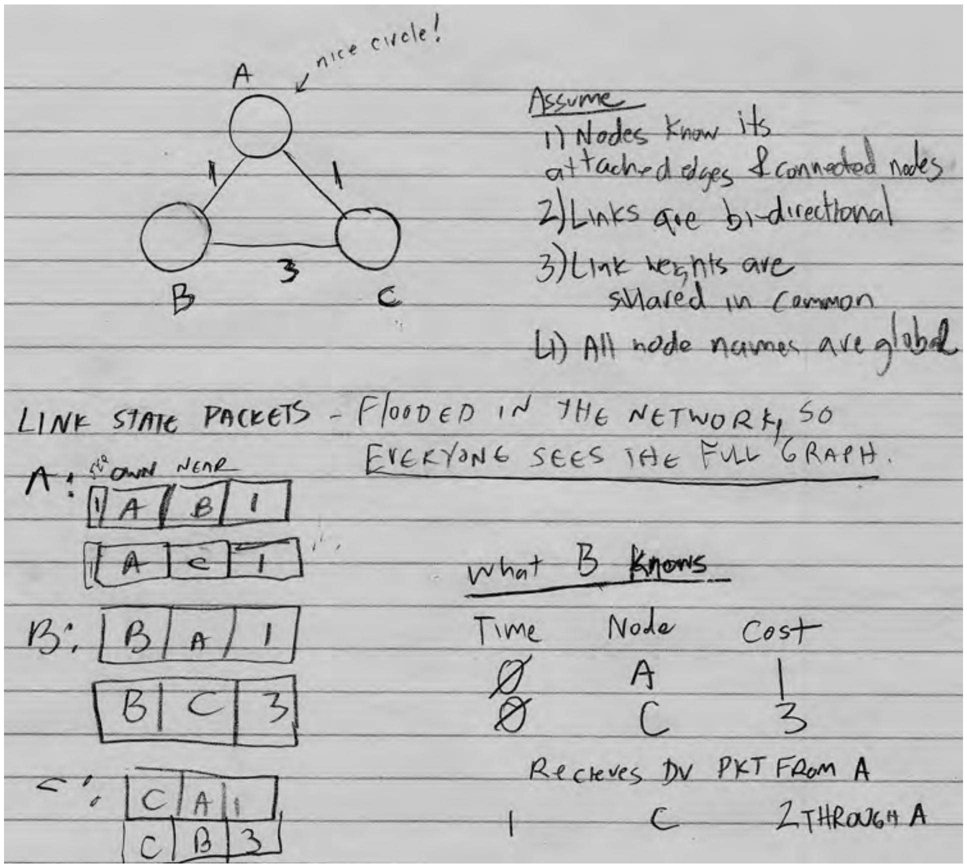


Figure 37: Packet Fragmentation

4.5.2 Link State Packets

A link-state routing protocol is one of the two main classes of routing protocols (the other is the distance-vector routing protocol). The basic concept of link-state routing is that every node **constructs a map** *of the connectivity to the network, in the form of a graph, showing which nodes are connected to which other nodes. Nodes then independently compute their shortest path to other nodes and build their routing tables*: `http://en.wikipedia.org/wiki/Link-state_routing_protocol`.

This graph or map is created using **link state packets**, packets with information about a routers neighbors. These packets are flooded within the network so that routers within a network can discover the network topology. They are used to identity new neighbors, whether link failures occur and update weights on congested links.



nice circle!
A
B
C
1
1
3
Assume
1) Nodes know its attached edges & connected nodes
2) Links are bi-directional
3) Link weights are shared in common
4) All node names are global
LINK STATE PACKETS - FLOODED IN THE NETWORK, SO EVERYONE SEES THE FULL GRAPH.
A: own neig
A B 1
A C 1
B: B A 1
B C 3
C: C A 1
C B 3
What B knows
Time Node Cost
0 A 1
0 C 3
Recieves DV PKT FROM A
1 C 2 THROUGH A

Figure 38: Link State Packets

4.5.3 Routing Tables

Table that maintains a list of "which one of my outgoing ports should I send this packet given its destination?". The table maintains a list of (DEST, NEXTHOP) pairs. Geographical locationing of addresses and sub netting allow these tables to shrink significantly. In the core, they are still relatively large (≈80,000 entries): but without those two concepts, we would need billions of entries.

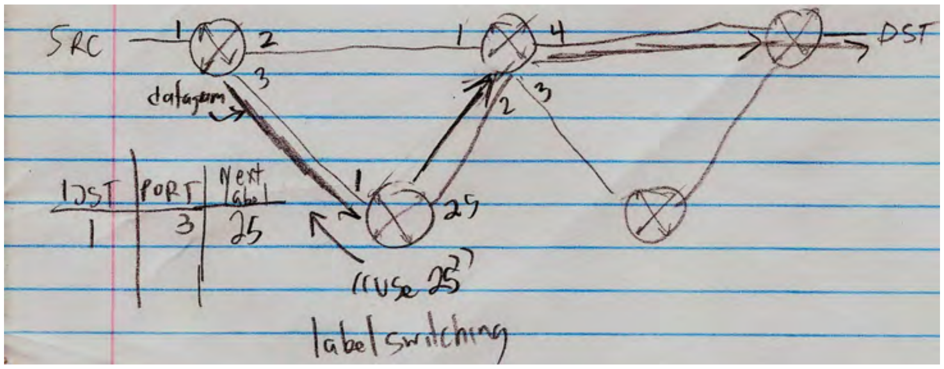
4.5.4 Building Routing Tables Via Shortest Path Routing

If we want to send traffic between two nodes in a network, a good idea would be to find the "shortest path" between those two nodes to send the traffic on. Shortest does not necessarily mean the least number of hops, but rather the path with the lowest end to end delay between the two nodes (this delay represents the total time to send between nodes including router processing time, propagation delay, queueing delay, congestion delay etc). If we can represent the delay on links in a network as weights, then we can use Dijsktras shortest path algorithm, or as Prof Keshav calls it, the "blob that ate everything algorithm". OSPF is an open source instantiation of Dijsktras.

I will not cover Dikskras algorithm here, as it is contained in every useful algorithms and networking book written, and extensive examples can be found online.

4.6 Virtual Circuits

Virtual circuits are another way to route packets, as opposed to IP. In most networks, IP won. Virtual circuits were a mistake, but are still used in some networks, such as cell phone networks.



DST	PORT	Next label
1	3	25

Figure 39: Virtual Circuit Path Setup

4.7 LANS

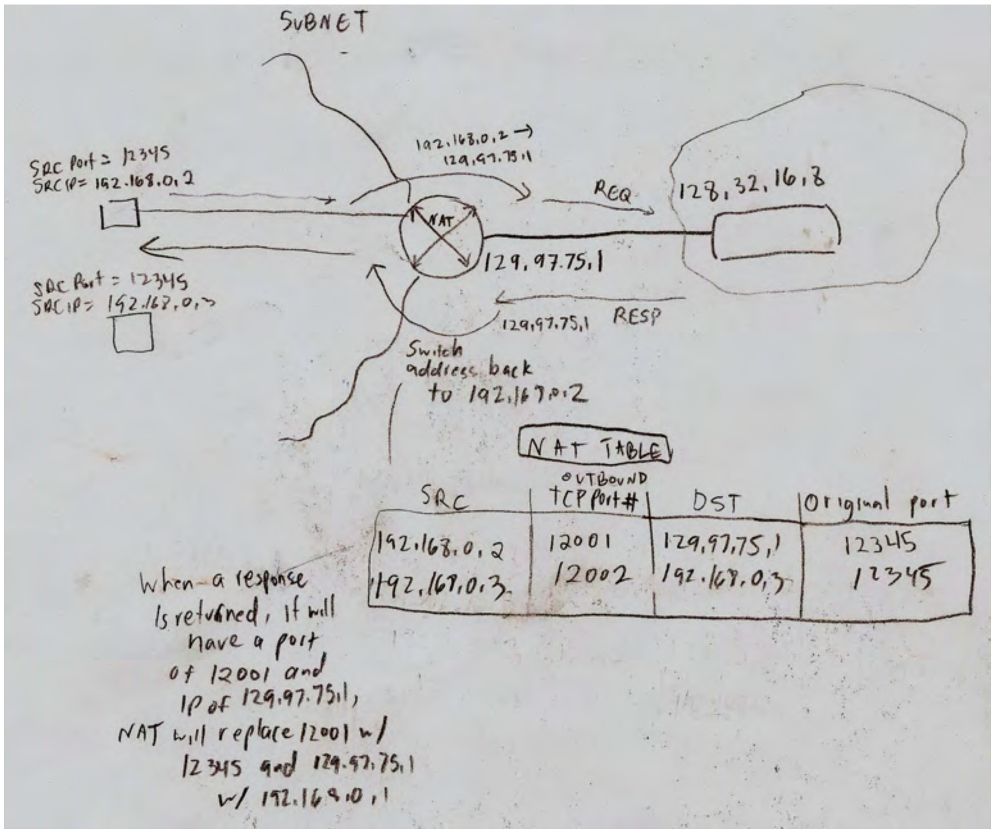
4.7.1 Private IP Addresses

These are not routable on the internet. You can use these in your private network

- 10.*/8 (DOF Arpa network founded the first network - the 10 network. Its number was later "retired")
- 192.168.*/16

4.7.2 Network Address Translation (NAT) and Port Address Translation (PAT)

The combination of these allow many computers in a network to share one external IP. Look up Translation of the Endpoint here: http://en.wikipedia.org/wiki/Network_address_translation



NAT TABLE

SRC	OUTBOUND TCP Port #	DST	Original port
192.168.0.2	12001	129.97.75.1	12345
192.168.0.3	12002	192.168.0.3	12345

When a response is returned, it will have a port of 12001 and IP of 129.97.75.1, NAT will replace 12001 w/ 12345 and 129.97.75.1 w/ 192.168.0.1

Figure 40: NAT table example

5 Link Layer

5.1 Main Purpose

Now we focus on actual physical links (99% of all links are Ethernet links). The link layer provides communication between nodes on the same link.

5.2 MAC Addresses

Every NIC has a unique hardware identifier, known as a MAC. MAC = Medium Access Control - the IP of the link layer. The addresses are organized as follows. 3 bytes for manufacturer ID, 3 bytes for a serial number (48 bits = 2^{48} MACs available.) This is how MAC addresses are guaranteed to be unique. Each manufacturer has their own unique ID, and they enforce uniqueness via serial numbers.

5.3 Link Properties

There are two types of "links"

- Point to Point: one actual link
- Broadcast medium: we have a hub with several links connected. Whenever a packet comes in on any of the links on the hub, it is forwarded on all other links (outgoing). Each device listens and if a packet with its MAC arrives, it keeps the packet, and otherwise drops it.

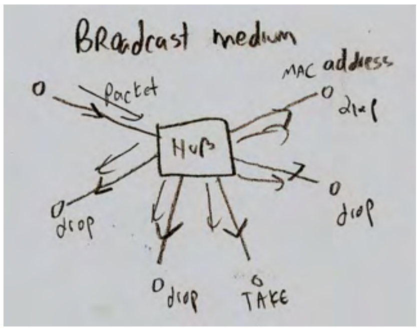


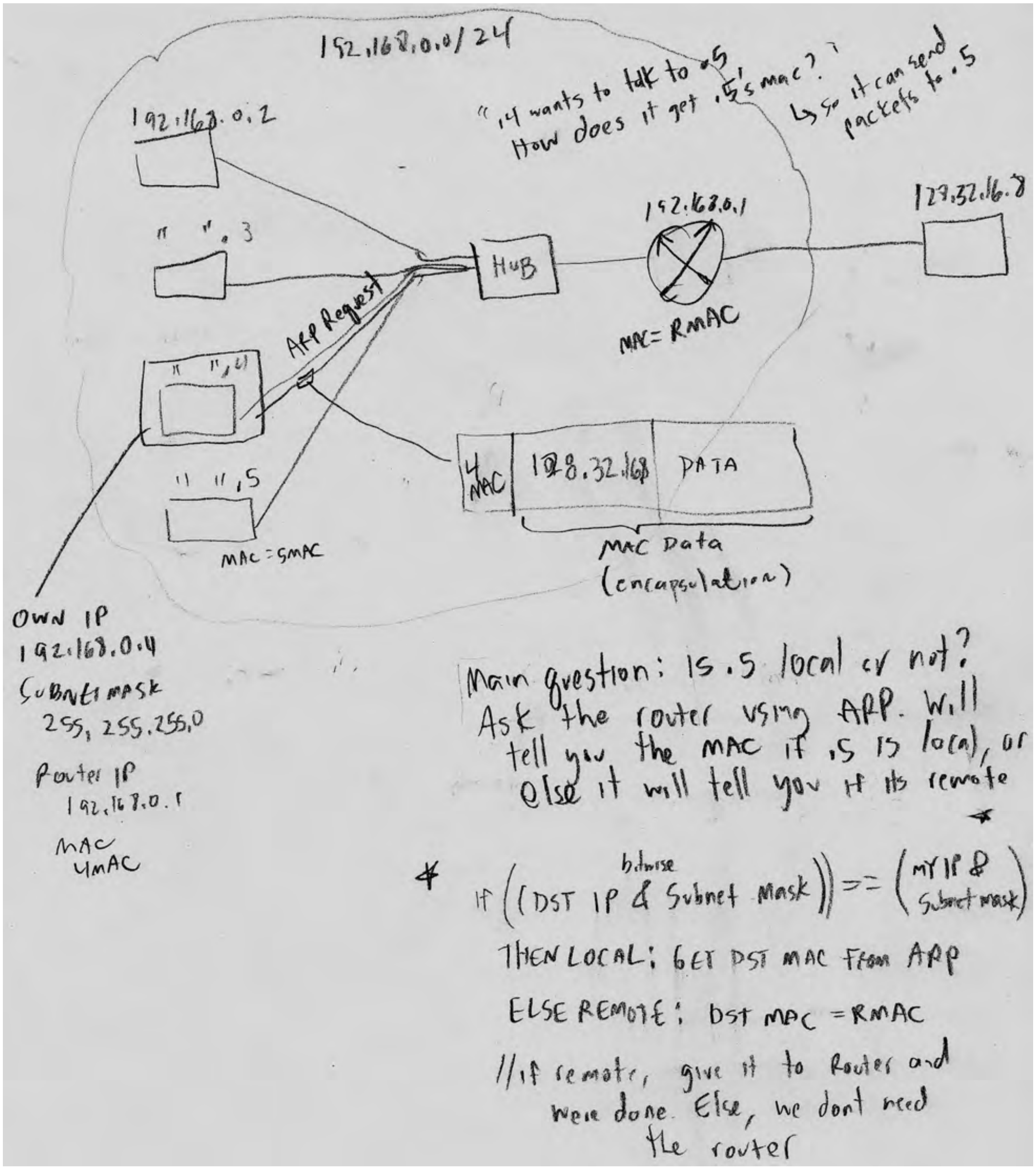
Figure 41: Broadcast Medium

There is a tradeoff between the following factors. You can't have them all!

1. Reliability:
   - Error Rate - rate which bits are changed on their way down the pipe.
   Can be solved using redundancy (Error correcting codes). Problem: determining least amount of information possible that must be sent to ensure redundancy. See information theory.
   - Packet Loss Rate
2. Throughput - how many bits per second can I send down the pipe (and have it be delivered on the other side). Keshav: bandwidth is a misuse of the term throughput, they are not synonymous.
3. Delay - if I put a packet in the pipe, how long will it take for it to be delivered? Humans are very sensitive to voice and video delays.
   - Speed of light (propagation delay). Keshav: "The speed of light isn't fast enough for communication".
   - Queueing delay
   - Link throughput - speed to send [large] files limited by size of pipe
4. Security/Privacy - is this link secure (is there encryption)?
5. Cost - $, energy (think of wireless links that drain batteries).

5.4 Address Resolution Protocol (ARP)

ARP is used to discover the MAC addresses of end hosts when only their IP is known. It is used for delivering packets within a LAN (see broadcast medium above). Consider a host A wants to send a packet to host B. If A and B are in the same LAN, if A knows B's MAC address, it can broadcast out the packet using B's MAC. If A and B are on different networks, A will send the packet via the internet to the router bordering B's network, and then B's router will then broadcast it on B's network using B's MAC address. In the first case, if A only know's B's IP, then it must broadcast a message on the network stating "I need B's MAC", the protocol known as ARP. When B hears this broadcast, B replies to A with its MAC. In the latter scenario, the router bordering B's network will use ARP similarly to determine B's MAC; A gives the packet to its local router and that's it.



192.168.0.0/24
192.168.0.2
" ".3
" ".4
" ".5
MAC=5MAC
".4 wants to talk to .5
How does it get .5's mac?"
↳ so it can send packets to .5
HUB
192.168.0.1
MAC=RMAC
128.32.16.8
ARP Request
4 MAC
128.32.168
DATA
MAC Data (encapsulation)
OWN IP 192.168.0.4
Subnet mask 255.255.255.0
Router IP 192.168.0.1
MAC 4MAC
Main question: is .5 local or not? Ask the router using ARP. Will tell you the MAC if .5 is local, or else it will tell you if its remote
if ((DST IP & Subnet Mask)) == (MY IP & Subnet mask)
bitwise
THEN LOCAL: GET DST MAC FROM ARP
ELSE REMOTE: DST MAC = RMAC
//if remote, give it to Router and were done. Else, we dont need the router

Figure 42: ARP

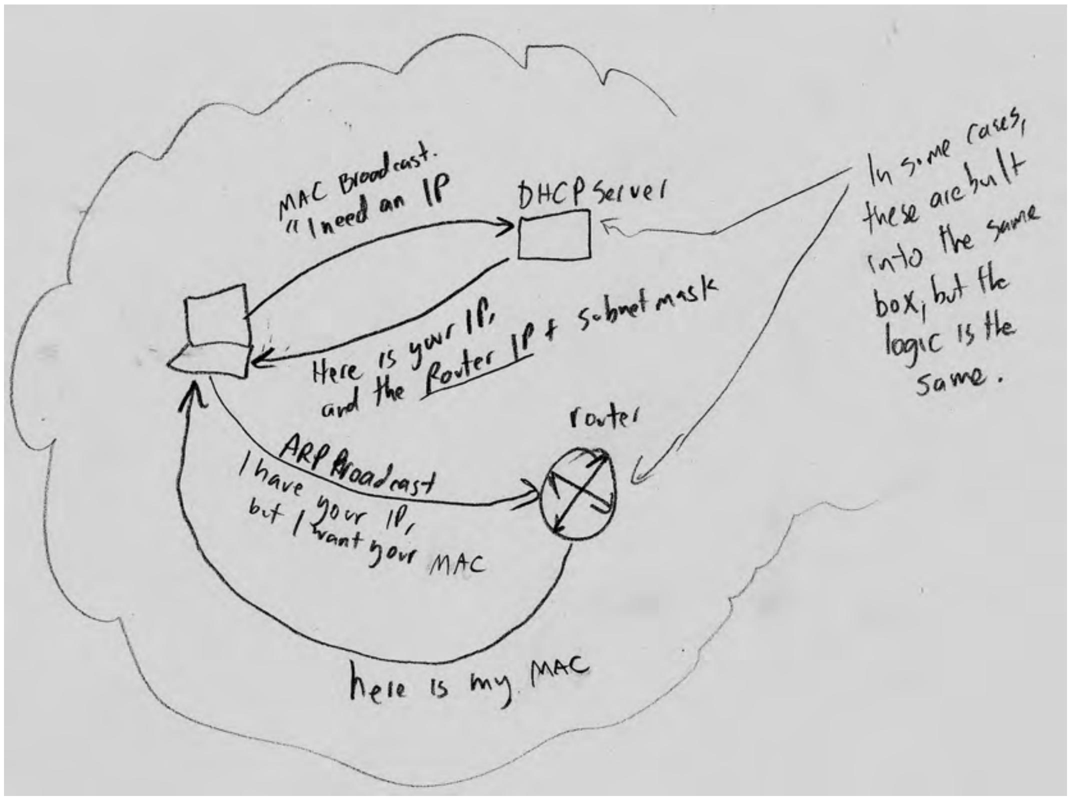

MAC Broadcast. "I need an IP
DHCP Server
In some cases, these are built into the same box, but the logic is the same.
Here is your IP, and the Router IP + subnet mask
router
ARP Broadcast I have your IP, but I want your MAC
here is my MAC

Figure 43: DHCP + ARP

5.5 Wifi

IEEE 802.11 {b,a,g,n,ac} standard

5.5.1 Components of Wifi

- Access point - has two network interfaces, one for the wireless connection, another for the wired connection
- NIC - your network interface card that can communicate with the access point.

Infrastructure mode (connecting to the access point), Ad Hoc mode (direct connection to another wifi device)

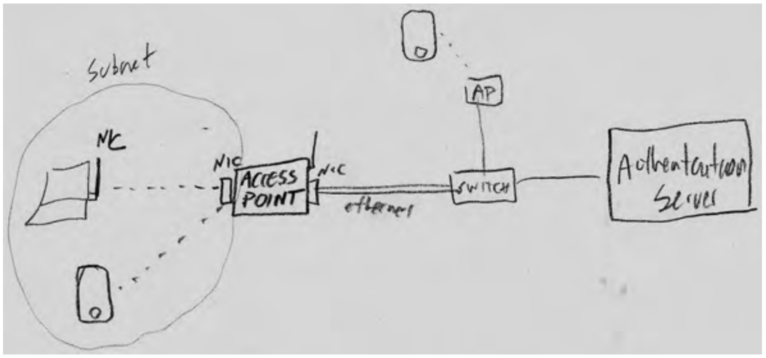


Figure 44: Components of WiFi

5.5.2 Wireless Links Aren't Really Links

- NOISY - Wireless links have loss rates as high as 90%. Turn your microwave on and watch your wifi connection!
- Channels (1,6,11). Wifi NICs have to be tuned to the same channel. If two devices submit a frame on the same channel at the same time, a collision will occur. AP NICs usually have multiple channels, such as 1,6,11. If three wireless devices are connected to these three different channels, the chance of collision is lower.
- Collisions are only detected at the receiver (access point), not at senders. In wireless, nodes cannot send and listen at the same time and collision detection is not possible.

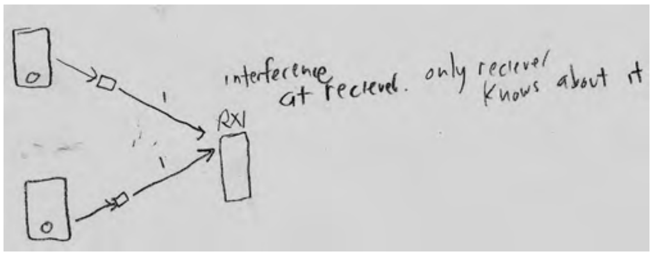


Figure 45: Collisions are only detected at the receiver

5.5.3 Wireless Mesh Networks

Keshav: "They will never work. Bad idea. Walk away. Not a valid research area".

5.5.4 Differences between b,a,g,n,ac

frequency*wavelength $= 3 * 10^8$m/s

- b: 11Mbps. Uses 3 simultaneous channels at the 2.4GHZ band, 1,6,11.
- a: 54Mbps. Uses 12 channels at the 5GHZ band. The problem is that water absorbs radiation at the 5GHZ band, so 80211a signals are constantly dropped reflected and absorbed by everything with water in it... Using the 700MHZ band is much better; wifi would travel much farther, not be absorbed by people, cats, metal... So why aren't we using 700MHZ band? Because governments decided to allocate this band to TV stations! Wifi got stuck using crappy 2.4 and 5GHZ bands. In the US and Canada, analog TV has been shut down. This freed up the 700MHZ spectrum. Now, it is up for auction. Guess who's buying it? Rogers, for their 4G network. Anywayssss, 2.4GHZ is better than 5GHZ (because just about everything has water in it), so a is rarely used.
- g: 56Mbps, 3 channels, 2.4GHZ.
- n: 108Mbps, 1 channel, 2.4GHZ.
- ac: more goodies, better rates.. its coming.

5.5.5 Security

When a wireless device detects an access point (AP), and wishes to connect to that AP, if the AP hosts a secured network then the device submits its credentials to the AP. The AP then verifies the credentials to access the network.

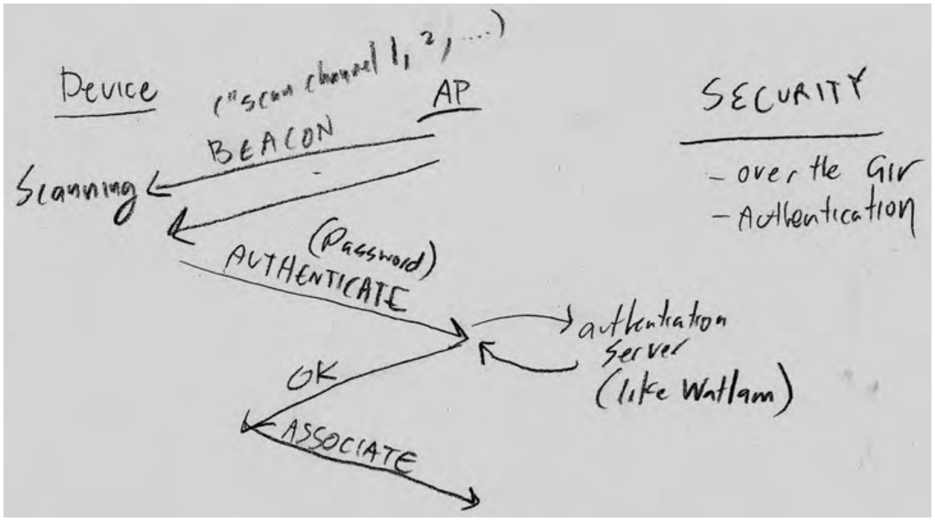


Figure 46: WiFi network association

5.5.6 CSMA/CA

CSMA/CA is the most common wireless transmission protocol. Senders wishing to transmit on a wifi link first inspect the channel to see if its idle. If the link is idle then the sender can transmit. If the link is not idle, then the sender waits a random period of time before transmission. This process then repeats. More information can be found here: http://en.wikipedia.org/wiki/Carrier_sense_multiple_access_with_collision_avoidance

5.6 Cell Phones

Keshav: "I'm so rich I don't have cell coverage at home", referring to the rich cats that live in the hilly part of San Francisco where there is no cell coverage.

5.6.1 Components

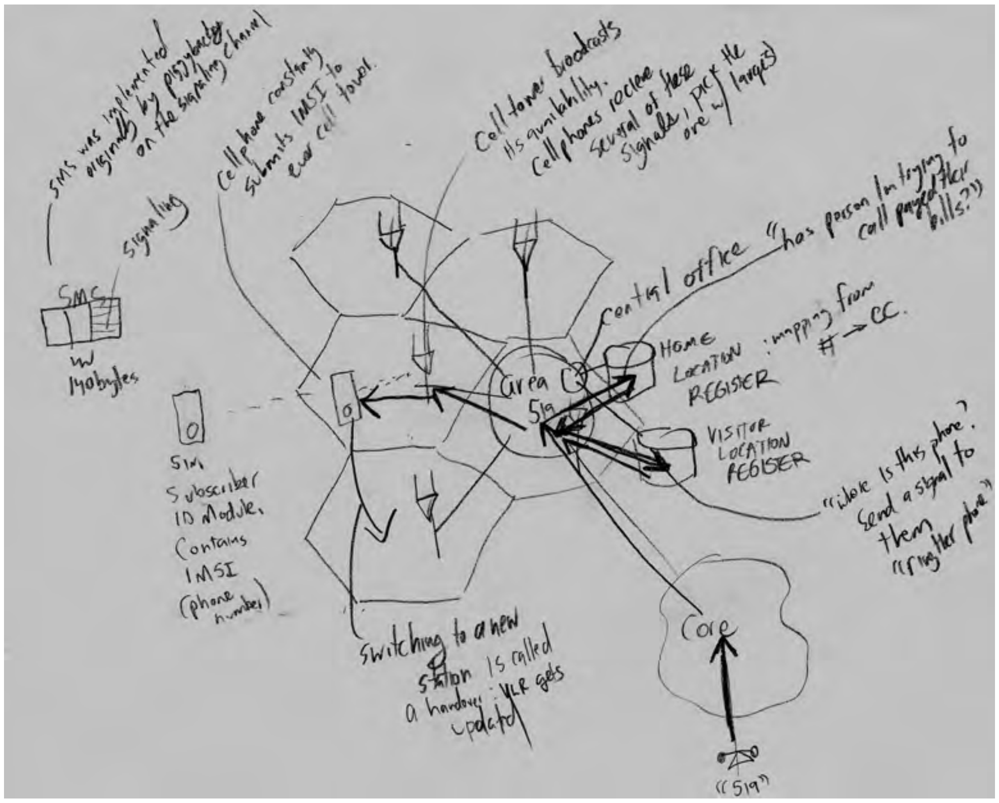

SMS was implemented originally by piggybacking on the signaling channel
signaling
SMS
w/ 140bytes
Cellphone constantly submits IMSI to ever cell tower.
Cell tower broadcasts its availability. Cellphones recieve several of these signals, pick the one w/ largest
central office
"has person Im trying to call payed their bills?"
HOME LOCATION REGISTER : mapping from # → CC.
VISITOR LOCATION REGISTER
"where is this phone? Send a signal to them "ring the phone"
Area 519
SIM Subscriber ID Module, Contains IMSI (phone number)
Switching to a new station is called a handover: VLR gets updated
Core
"519"

Figure 47: Cell network components and layout

5.6.2 Cell Concept

The reason the above cellular structure is used (note that in real life cells are not nice honeycombs, they look more like amoebas) is to share frequency:

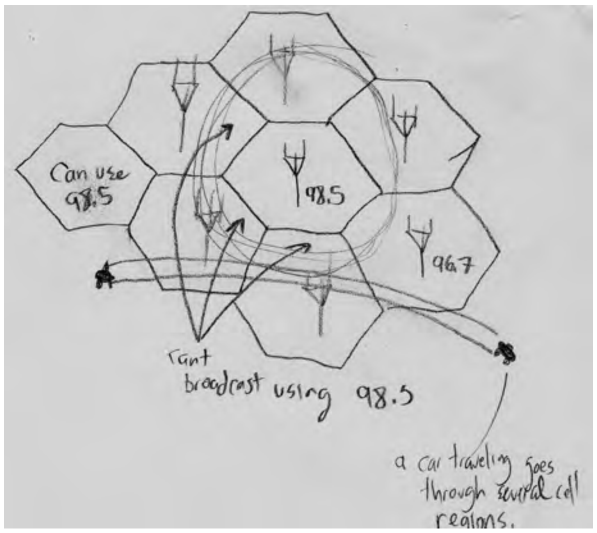

Can use 98.5
98.5
96.7
cant broadcast using 98.5
a car traveling goes through several cell regions.

Figure 48

5.6.3 Generations

1. 1G: Analog. Problem is that when you drove into a coverage area, if no frequencies were left, your call would be dropped. Another problem is that anyone can hear anyone else's calls by tuning their phone.
2. 2G: Digital. FDM (frequency division multiplexing) + TDM (time division multiplexing), GSM. Base station sends out a time signal. If you are in "slot 4" for example, its your turn to transmit.
3. 3G: Data Oriented. Peak data rates = 200kbps
4. 4G: "mobile broadband, LTE (long term evolution)". All IP based (IPV6). Peak data rates = 1Gbps down, 500Mbps up.

5.6.4 Roaming

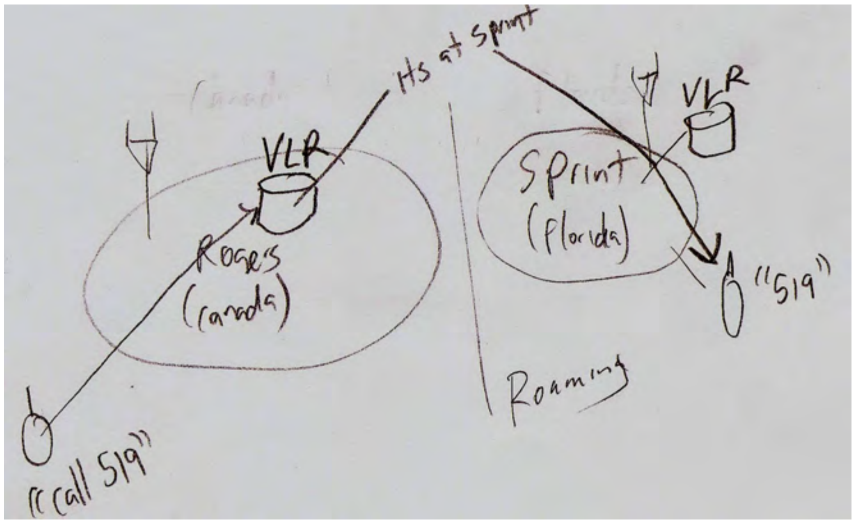


Figure 49: Roaming

6 Security

We are in the age of hacking!

6.1 Attacks

- Copy messages
- Inject messages
- Replace/Modify messages
- Spoof messages
- Infer messages
- DoS attack

6.2 Goals

Despite the presence of malicious attacks, we want to ensure

- Privacy: messages can't be eavesdropped or infered
- Authentication: messages sent to right party
- Integrity: messages can't be tampered with
- DoS: ensure delivery

6.3 Encryption

Encode messages such that people without the necessary *secret keys* cannot read the message.

6.3.1 How Secure Is Encryption?

- Attacker could try all keys. In this case strength of crypto depends on number of possible keys
- Breaking encryption should be exponential in key length

6.3.2 How Practical Is Encryption?

- Usability depends on encryption/decryption to be efficient
- Security depends on long (hard to guess) keys. Too bad humans aren't good at remembering these!
- Suppose we have a 5 letter password that can be composed of only lowercase letters. There are $26^5 \approx 10M$ possible passwords. In 1975 this took one day to break, in 1992 it took 10 seconds, and in 2008, it took a millisecond.
- Suppose now we have 6 letter password that can be composed of upper, lower, numbers, control characters. Now we have $70^6 \approx 600B$ possible passwords. In 2008, with a 1000 parallel machines, this could be broken in less than one second.

6.4 Symmetric Key Encryption

- "Secret Key Crypto". There is a single private key for encrypt/decrypt.
- Problem: both parties have to know the key

6.4.1 One Time Pad: Most Secure Encryption Algorithm Ever

A sequence of random bits the same sequence or our message. XOR the message and the pad bits together to get the cipher text. Decrypter does the same to get the original plaintext. Discard pad after one time use.

6.4.2 DES: Data Encryption Standard (although not anymore...)

Key is 56 bits. Message is split into chunks. Chunks are scrambled together (cipher block chaining), then encrypted.
Note that AES has taken over this standard.

6.5 Public Key Cryptography

- Keys for encrypting are public (K_A^+, K_B^+)
- Keys for decrypting are private (K_A^-, K_B^-)
- Alice wants to send m to Bob.
$$c = K_B^+(m)$$
- Bob decrypts by
$$m = K_B^-(c) = K_B^-(K_B^+(m))$$
- Public Key Crypto relies on "one way functions", functions that are non-invertible. For example $F : m \to c$. If we know c, we should **not** be able to figure out m by applying some known inverse F'.

6.6 RSA (1978)

Two couponenets

1. Choice of public/private keys.
2. Choice of encryption algorithm

In RSA, K_B^+, K_B^- are chosen by:

1. Choose two large prime numbers p and q
2. Compute $n = pq$ and $z = (p-1)(q-1)$
3. Choose a number $e < n$ and e and z are relatively prime (no common factors other than 1).
4. Find a number d such that $ed - 1$ is divisible by z, or equivalently $ed - 1 \mod z = 0$, or equivalently $ed = 1 \mod z$, or equivalently $ed \mod z = 1$.
5. $K_B^+ = (n, e)$
6. $K_B^- = (n, d)$

Now suppose Alice wants to send m to Bob. We'll assume $|m| < n$. To encrypt, she computes $c = m^e \mod n$. To decrypt, Bob computes $m = c^d \% n$.

6.6.1 Example

Example: A wants to send a message to B. Bob chooses $p = 5$ and $q = 7$ as his two primes.
ENCRYPT Then $n = 35$ and $z = 24$. We can choose $e = 5$ and $d = 29$. Let "M = 'L'", thus $m = 12$. $c = m^5 \mod 35 = 17$.
DECRYPT $m = 17^{29} \mod 35 = ... \mod 35 = 12 =' L'$

6.6.2 Why Does RSA work?

Theorem: if p, q prime and $n = pq$, then $x^y \mod n = x^{(y \mod (p-1)(q-1))}$

$$\begin{aligned} K_B^-(K_B^+(m)) &= (m^e)^d \mod n \\ &= m^{(ed \mod (p-1)(q-1))} \mod n \quad //(ed \mod (p-1)(q-1)) = 1) \\ &= m \mod n \quad \text{since } m < n \end{aligned}$$

Breaking RSA is as hard as factoring.

6.7 Message Integrity

To authenticate a message, Bob must verify two things. If both of these are verified, then m has integrity.

1. m really did originate from Alice
2. m was not tampered with in transit

6.7.1 Cryptographic hash function

Has the following properties

- Computationally infeasable to find two messages X and Y such that $H(X) = H(Y)$ ("collision").

6.7.2 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 the hashes match, every things OK.

ATTACK: Trudy creates m', interrupts $m, h(M)$ and instead sends $m', H(m')$. Now Bob receives a different message!

6.7.3 Attempt 2: shared secret

Authentication key s. Integrity protocol:

1. Alice computes $H(m + s)$
2. Sends $(m, H(m + s))$ to Bob
3. Bob checks if $H(m + s)$ matches; if so OK.

ATTACK: Trudy can always change m, but it can be detected. She can still shut down the communication essentially.

6.7.4 Digital Signatures

Attests than an entity owns something or agrees to its contents. Needs to be:

- verifiable
- non-forgable

Bob's signature must be unique. In Public key crypto, Bob has unique private and public keys.
Bobs signature for m is $K_B^-(m)$. We are using the keys in reverse so that anyone can verify Bob's signature but only Bob can sign it.
ATTACK: Trudy announces K_B^+ as her public key signature, essentially stealing Bob's unique public key.

6.7.5 Public Key Certification

Certifies that a public key belongs to a specific entity. Needs to be a 3rd party organization. They verify people are who they say you are an issue certificates that bind your identity to your key. If you want to have a digital signature, you have to pay someone for it.

6.8 End-point Authentication

6.8.1 ap1.0

Process of proving your identity. Authentication protocol 1.0, a p1.0. Says "I'm Alice".
ATTACK: Trudy: "I'm Alice"

6.8.2 ap2.0

Alice says ("I'm Alice", IP address)
ATTACK: Trudy spoofs Alices IP

6.8.3 ap3.0

Alice says ("I'm Alice", password)
ATTACK: Trudy can sniff the traffic on the network and just grab the password

6.8.4 ap3.1

Alice says ("I'm Alice", encrypt(password))
ATTACK: Trudy can record this message and then later resend it to the server: she would be authenticated into the server without ever knowing the password. This is known as a playback attack.

6.8.5 ap4.0

1. Alice says "I'm Alice"
2. Bob chooses a one-time use number R and sends R to Alice
3. Alice encrypts R using a K_{A-B} (a shared key between A and B) and sends $K_{A-B}(R)$ to Bob
4. Bob decrypts, verifies R

6.9 In Practice

No way to prove secureness. You can show known attacks don't work (still not a proof of secureness). For example you can hire some "white hat" hackers to hack your system, find the flaws, and then fix them. You can also do "vulnerability scanning"; scanning for known problems in your system. You can check for open unsecured ports, try to guess for simple authentication passwords, social engineering,... Essentially trying to break or hiring experts to try to break your system is the current industry standard.

7 Distributed Systems

Class definition: *set of interacting components with well-defined interfaces with specific inputs and outputs and provides functionality*

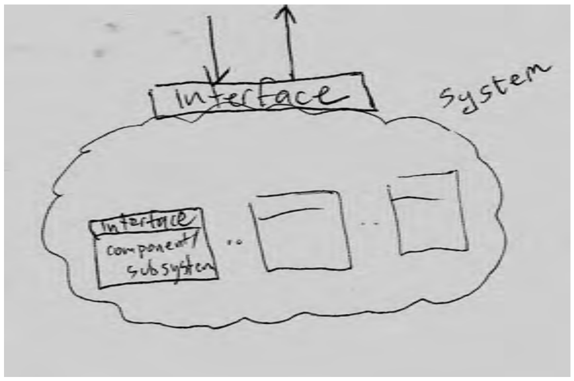


Figure 50: View of a System

Distributed systems are really networked applications:

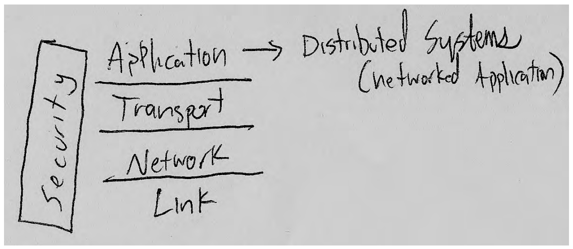


Figure 51: Roaming

Computer systems have *resources*:

1. Computation
2. Storage
3. Communication

7.1 Motivation

Centralized systems...

- don't scale
- are not fault tolerant (single point of failure)
- Don't provide delegation (local control; think of google.ca/google.com)

7.2 Concurrency Issues

The main implementation problem of distributed systems is concurrency. Imagine you have a website that sells stuff. The number n available of item X is stored in a database. Suppose two customers come to the website. The first asks "how many X can I purchase", to which the response is n. Suppose that while the first customer is still in this transaction and hasn't purchased anything yet, customer 2 comes along and asks the same question. What is the best response? Should you return 0 thinking that the first customer will purchase the items, or n thinking the first customer will not?

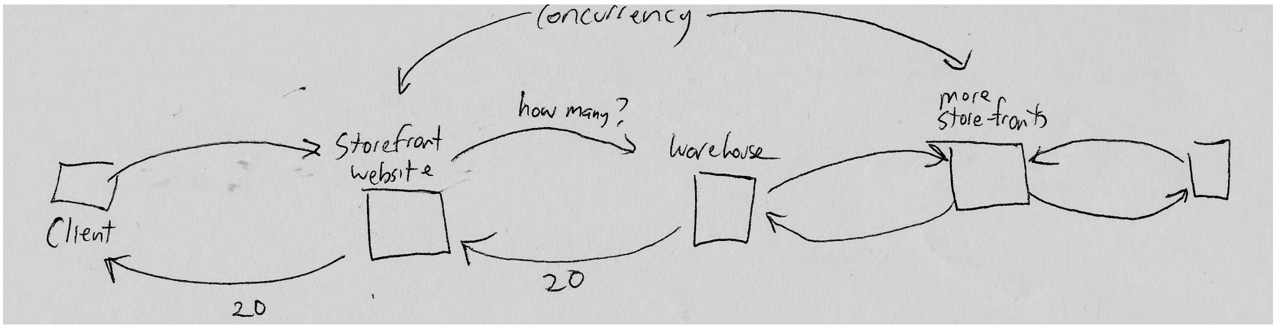


Figure 52: Concurrency 1

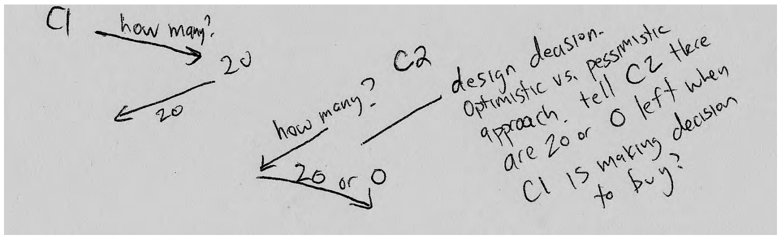

C1
how many?
20
20
C2
how many?
20 or 0
design decision- Optimistic vs. pessimistic approach. tell C2 there are 20 or 0 left when C1 is making decision to buy?

Figure 53: Concurrency 2

7.3 Desirable Properties

The first 4 are known as the ACID properties and come from the database community

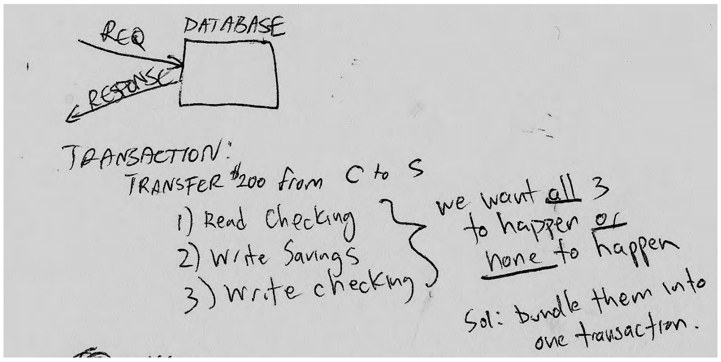

REQ
DATABASE
RESPONSE
TRANSACTION:
TRANSFER $200 from C to S
1) Read Checking
2) Write Savings
3) write checking
we want all 3 to happen or none to happen
Sol: bundle them into one transaction.

Figure 54: Bank Database Example

- **A**toxicity: atomicity is the property of transactions:
- **C**onsistency: after a transfer, checkings + savings is same. In this database example atomicity ensures consistency, but this is not always the case.
- **I**solution: things don't collide with each other. If I am doing a transaction on my account and someone else is doing a transaction on their account, these two transactions should not affect each other. This problem occurs in networks all the time; if too many people are accessing a server at the same time, and the server crashes, then everyone is affected. This is a loss of isolation. This is almost never achievable.
- **D**urability: once somethings happened it "stays happened". If you make a transaction today, when you check tomorrow the transaction preformed should not have been undone...
- Fault tolerance: link failures, computer failures, disk failures, packet corruption, incorrect computation (very hard to correct or detect), disk/memory corruption
- Scalability: we want things to grow without bound (impossible, but we can get close)
- Synchrony: different components of the DS agree on time and what happened before. "Happens before" relationship.
- Transparency: implementation detail. The services provided by the system should be independent of the implementation. The end user should not have to be concerned with/ should not care about how the system is implemented. Furthermore, if the implementation changes, the user should not be able to tell the difference.
- Performance: usually based on some metrics: throughput, time between failure, time between delay. Metrics depend on what applicable faults are.
- Heterogeniety: We have an implementation where we do not need to use all of the same thing. For example we should be able to build a network where the brands of routers in the network can all be different.

7.4 Locking

Concurrency + Sharing gives a need for locking. Suppose we have two processes that are trying to access some shared variable or state. **Locking**: the first process that needs to edit the shared state, it acquires the key to the lock. No one else may edit the shared state while that process has the key. When the process is done it releases the key for other processes to edit the shared state. The shared state in this context is known as a *critical section.*

7.4.1 Locking Principles

- Many readers
- 1 writer many readers: "reader writer lock". Use a queue manager!

7.4.2 What could go wrong?

1. Get a lock in a process, and the process dies. At this point everyone else can't do anything. Solution: lock manager can keep a timer. If the timer expires, it can check to see whether the process is still alive or not and then release the lock if not.
2. Deadlock. Happens when two processes each hold a lock that the other process needs. Neither process can proceed because they are each waiting for the other to release. Solution: two phase locking. Acquire all locks before we use them, and have a specific order that all threads/processes follow. Heres a deadlock example from the Java website: http://docs.oracle.com/javase/tutorial/essential/concurrency/deadlock.html

Note there is a set of conditions required for a deadlock to occur called the "Coffman Conditions". More info here: http://en.wikipedia.org/wiki/Deadlock

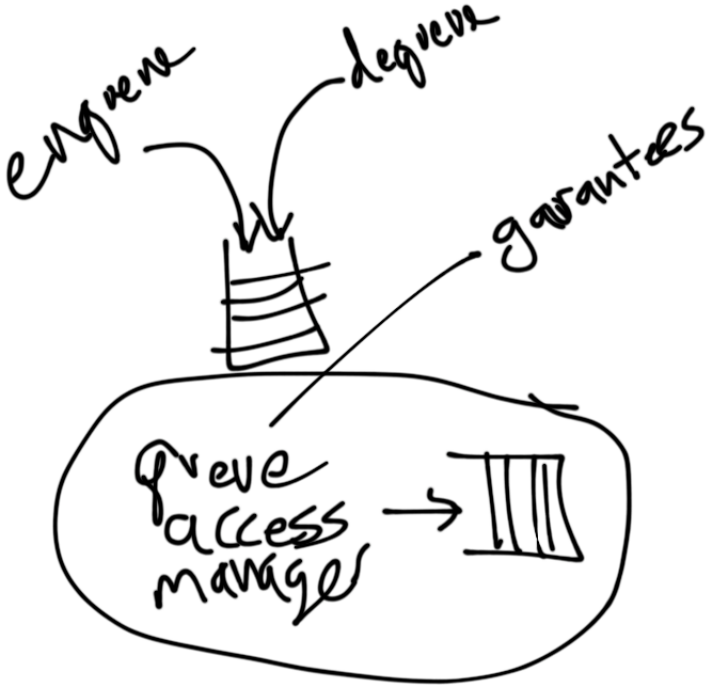


Figure 55: Queue Manager

7.4.3 Implementation

Java/C#: Synchronized objects: http://docs.oracle.com/javase/tutorial/essential/concurrency/sync.html
LOCK MANAGER - pthreads
Messages: send a message to some process that returns the information or writes the information you want.

7.5 Design Principles

- **No global state.** Maintain distributed state; the state of the system is the state of all of the subsystems. Each subsystem maintains their own local state, and should be able to make actions without knowing about the state of all other subsystems.
- **No central controller.** For example, one central global lock manager. This doesn't scale.
- **No global clock (?)** Not clear if we should still use this principle. In the old days it was very difficult to keep time synchronized among many computers. Most DS research has assumed over the past decades that no one can tell time. Nowadays with GPS receivers in almost everything that keep time very well, this assumption may disappear. Another advancement is the invention of a time server; all computers in the system can access the time server to synchronize time.
- **Separate policy & mechanism (customizability).** For example, the notion of locking is a mechanism. However, how granular the locking should be (in some application) is not something the lock manager can decide. The policy means when you use the lock manager, then you choose the granularity. As another example, look at toolbars in modern OSs. The mechanism is the toolbar and it lets you put up to X things in it. However each user should be able to choose their policy; what stuff shows up in their toolbar.
- **Explicit interfaces.** For every subsystem, there is an interface that you must go through to access components within that subsystem. For example if there is some array in a subsystem, there should not be some pointer from another subsystem in that array. All accesses should go through the interface.

References

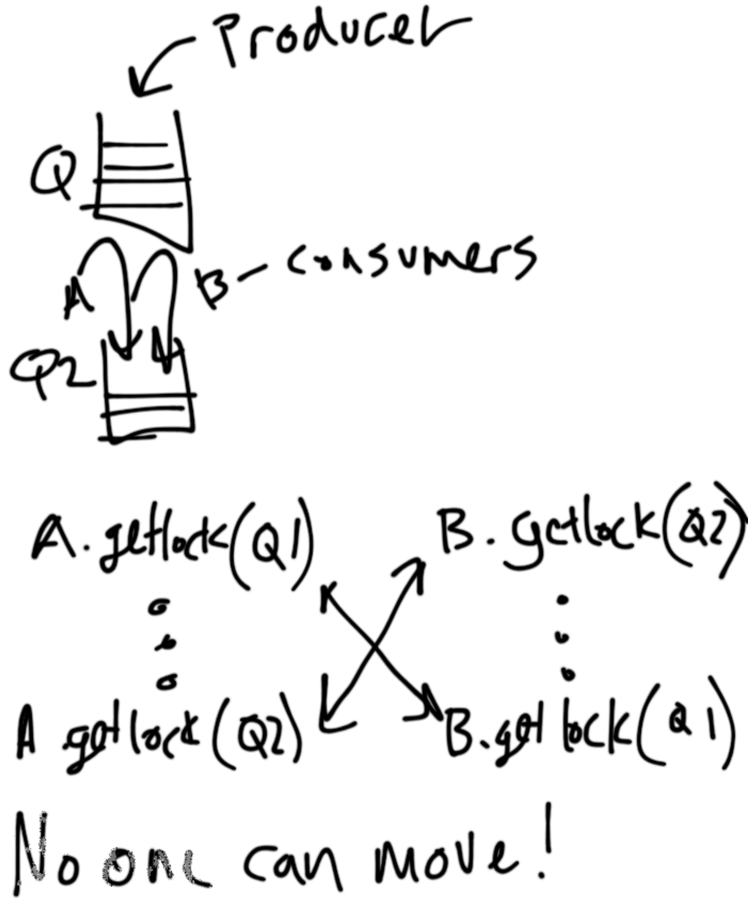


Figure 56: Deadlock example