How Does the Web Really Work?

Tom Anderson

Themes

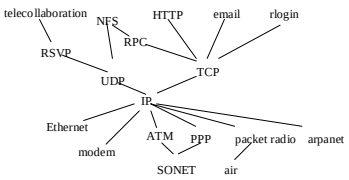
- Keep it simple => get everyone to participate
  - in biz lingo, "network effect"
- Murphy's law => fault tolerance/recovery
  - something always fails
- Scalability => no one in control
  - and not everyone is equally trustworthy

Protocols

- Protocol: agreement between two parties as to how information is to be transmitted
  - more valuable with more users ("network effect")
  - economic incentive to develop standards
    => lots and lots and lots of protocols
- Standardize protocols vs. standardize interfaces?

Layering

Build complex services on top of simpler ones



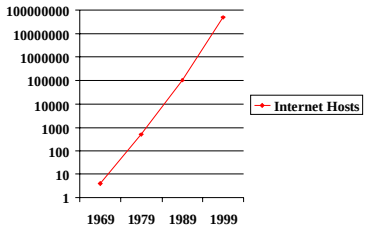
Internet -- network of networks

- Local network delivers packets
- Routers move packets between networks
- IP interoperability on top of any potential network or link layer
  - modem, Ethernet, token ring, cell phone, ADSL, cable modem, smoke signals, …
- Minimize requirements on underlying net

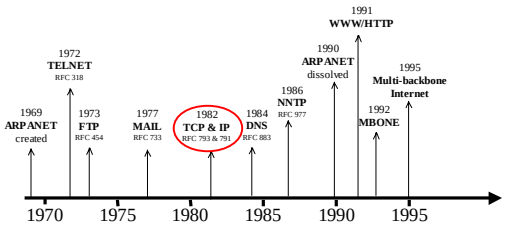
Internet Design Principle

- Assume smart hosts and dumb network
  - interoperate with *any* packet network
- End to end principle: put feature in network iff it can be completely implemented there (on any network)
  - routing => local network + Internet
  - reliable retransmission => end hosts
  - security => end hosts
  - congestion control => end hosts

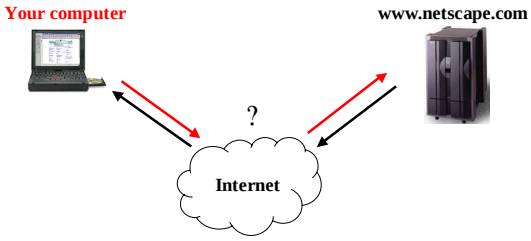
An Engineering Triumph



A brief Internet history...



What happens when you click on a Web link?



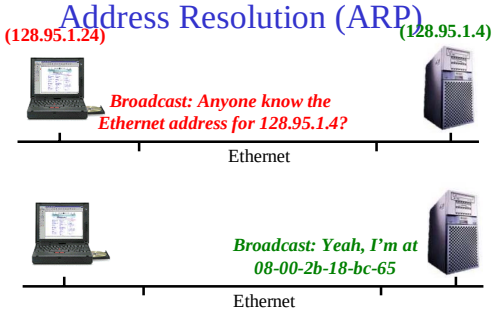
Different kinds of addresses

- Domain name (e.g. *www.netscape.com)*
  - Global, human readable
- IP Address (e.g. 207.200.73.8)
  - Global, works across all networks
- Ethernet (e.g. *08-00-2b-18-bc-65)*
  - Local, works on a particular network
- Q: why is Ethernet address longer than IP?

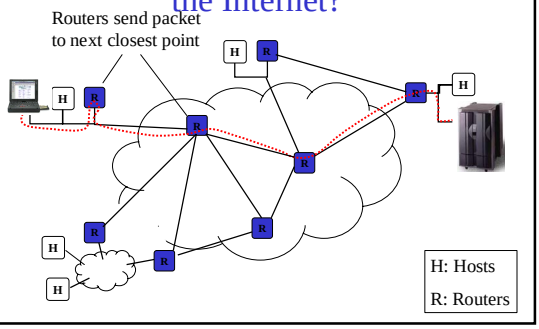
Finding the right IP address: Domain Naming System (DNS)

But netscape.com is really a distributed cluster of machines.
How do we return closest representative?

Finding the right Ether address: Address Resolution (ARP)



How does a packet get through the Internet?



How do the routers know where to send data?

- Forwarding tables at each router
  - original Internet: manual update
  - stops working if there are failures
- Automatic update based on "cost"
  - exchange tables with neighbors
  - use neighbor with smallest hop count
  - path grows from destination to source
- Self-healing on failure (if redundant paths)

Issues

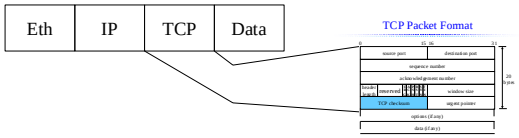
- Can we get into a loop?
- Router table size scales with Internet size?
  - Single table entry for range of addresses
- Everyone must use same routing algorithm?
  - meta-Internet: "autonomous systems"
- Best hop count != best route
  - What if a node says it has zero cost to everywhere?

Have address, now send data?

- Murphy's Law applies to networks
  - Data can be corrupted
  - Data can get lost
  - Data might not fit in a single packet
  - Data can be delivered in the wrong order
  - etc...

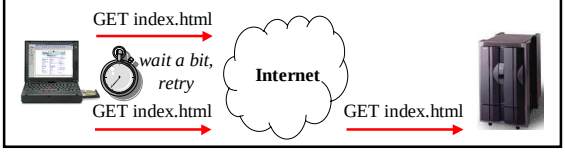
What if the data gets corrupted?

Solution: Add a *checksum*



What if the data gets lost?

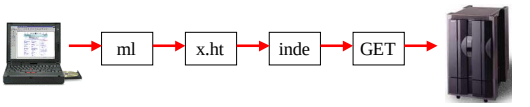
Solution: *Timeout* and *retransmit*



What if the data doesn't fit?

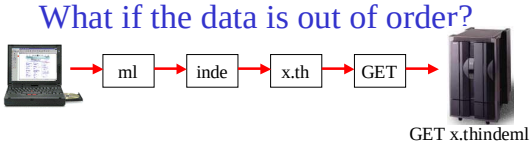
- On Ethernet, max IP packet is 1.5kbytes
- Typical web page is 10kbytes

Solution: *Fragment* data across packets



GET index.html

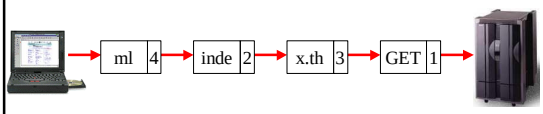
What if the data is out of order?



GET x.thindeml

???

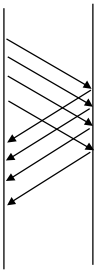
- Solution: Add sequence numbers



GET index.html

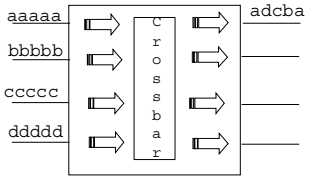
How do we keep the pipe full?

- Send multiple packets without waiting for first to be acked
  - window size = bandwidth * delay
- TCP sliding window
  - Send new packet when kth previous packet is acked
  - Sender keeps list of unack'ed packets; resends after timeout



What if network is overloaded?

- Rate of pkts arriving > rate of pkts departing



- Router will drop any overflow packets

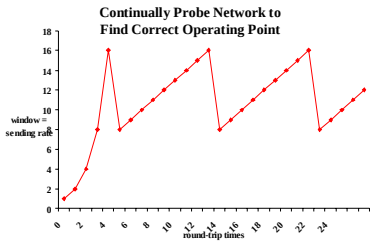
Congestion Collapse

- What happens as we add more users?

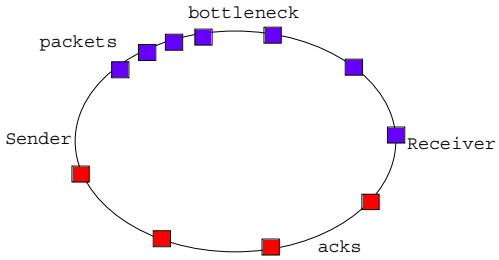
TCP Congestion Control

- Set retransmission timer to measured delay
- Infer network bandwidth from packet loss
  - drops => congestion => reduce rate
    - drops also caused by link noise!
  - no drops => no congestion => increase rate
  - additive increase/multiplicative decrease
- On startup, probe network to find bottleneck
  - "slow start" (exponential increase)

TCP Congestion Window



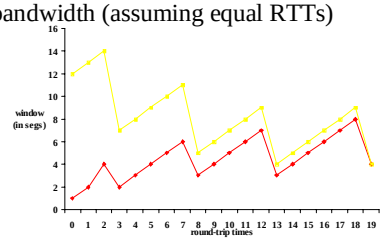
Avoiding burstiness: ack pacing



Window size = round trip delay * bit rate

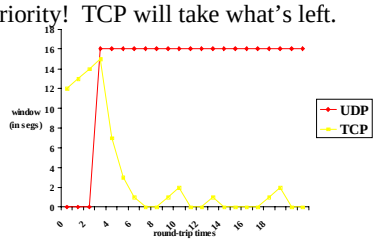
What if two TCP connections share link?

- Reach equilibrium independent of initial bandwidth (assuming equal RTTs)



What if TCP and UDP share link?

- Independent of initial rates, UDP will get priority! TCP will take what's left.

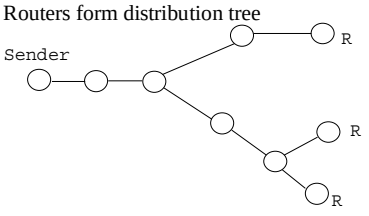


What if data has a deadline?

- Ex: multimedia, teleconferencing
  - original Internet: out of luck!
- To provide guarantees, need
  - admission control
  - resource management at routers
  - ex: telephone network has busy signals + explicit schedules at each switch
- How do we add this to the Internet?
  - Short term winner: priority queues

What if multiple receivers?

- Multicast
  - Send data only once down shared link
  - Routers form distribution tree



Issues

- Routing: how do we build tree?
  - Dynamic group membership; “broadcast and prune”
- Reliability: what if data is dropped?
  - Acks? would overwhelm sender
  - Naks? if drop is high in the tree -> ditto!
- Heterogeneity and congestion control
  - layered multicast

What if sender is malicious?

- Every packet has source, destination IP addresses
- But! Host can put *anything* in IP header
  - packet may have come from anywhere
  - firewalls to enforce sanity checks
    - ex: source must be from other side of wall
    - ex: only allow reply packets
  - encryption/digital signatures for authentication/privacy

Bottom Line

- No magic!
  - Keep it simple => get everyone to participate
  - Murphy’s law => fault tolerance/recovery
  - Scalability => no one in control

What Does the Future Hold?

- Moore’s Law: Exponential improvement
  - CPU speed (Mflops/$) 80%/year
  - DRAM memory (Mbytes/$) 60%/year
  - Disk storage (Gbytes/$) 100%/year
  - Network BW (Mbytes/sec/$) 60%/year
  - US GDP 3%/year
  - US population 1-2%/year
  - Software complexity 10-20%/year?

Time to Rethink?

1980’s Internet	2000’s Internet
Low bandwidth * delay	High bandwidth * delay
Low drop rates, < 1%	High drop rates, > 5%
Few, long-lived flows	Many short-lived flows
Every host a good citizen	TCP "accelerators" & inelastic traffic
Symmetric routes & universal reachability	Assymetric routes & private peering
Hosts powerful & routers overwhelmed	Hosts = toasters & routers intelligent?
Limited understanding of packet switching	ATM and MPP network design experience