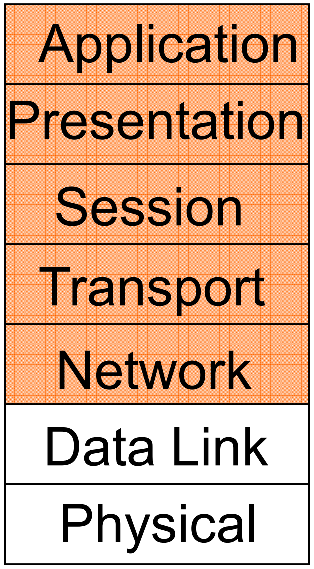
Potpourri

- Topics
  - Peer-to-peer
  - Firewalls
  - NAT
  - VPNs
  - NTP



Peer-to-Peer Systems

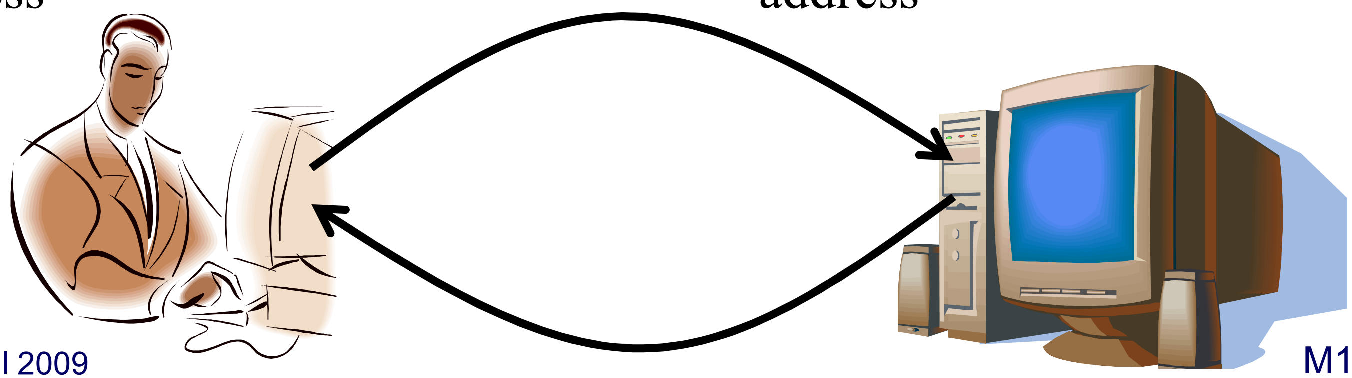
- Quickly grown in popularity:
  - Dozens or hundreds of file sharing applications
  - In 2004:
    - 35 million adults used P2P networks – 29% of all Internet users in USA
    - BitTorrent: a few million users at any given point
    - 35% of Internet traffic is from BitTorrent
  - Upset the music industry, drawn college students, web developers, recording artists and universities into court

- But P2P is not new and is probably here to stay

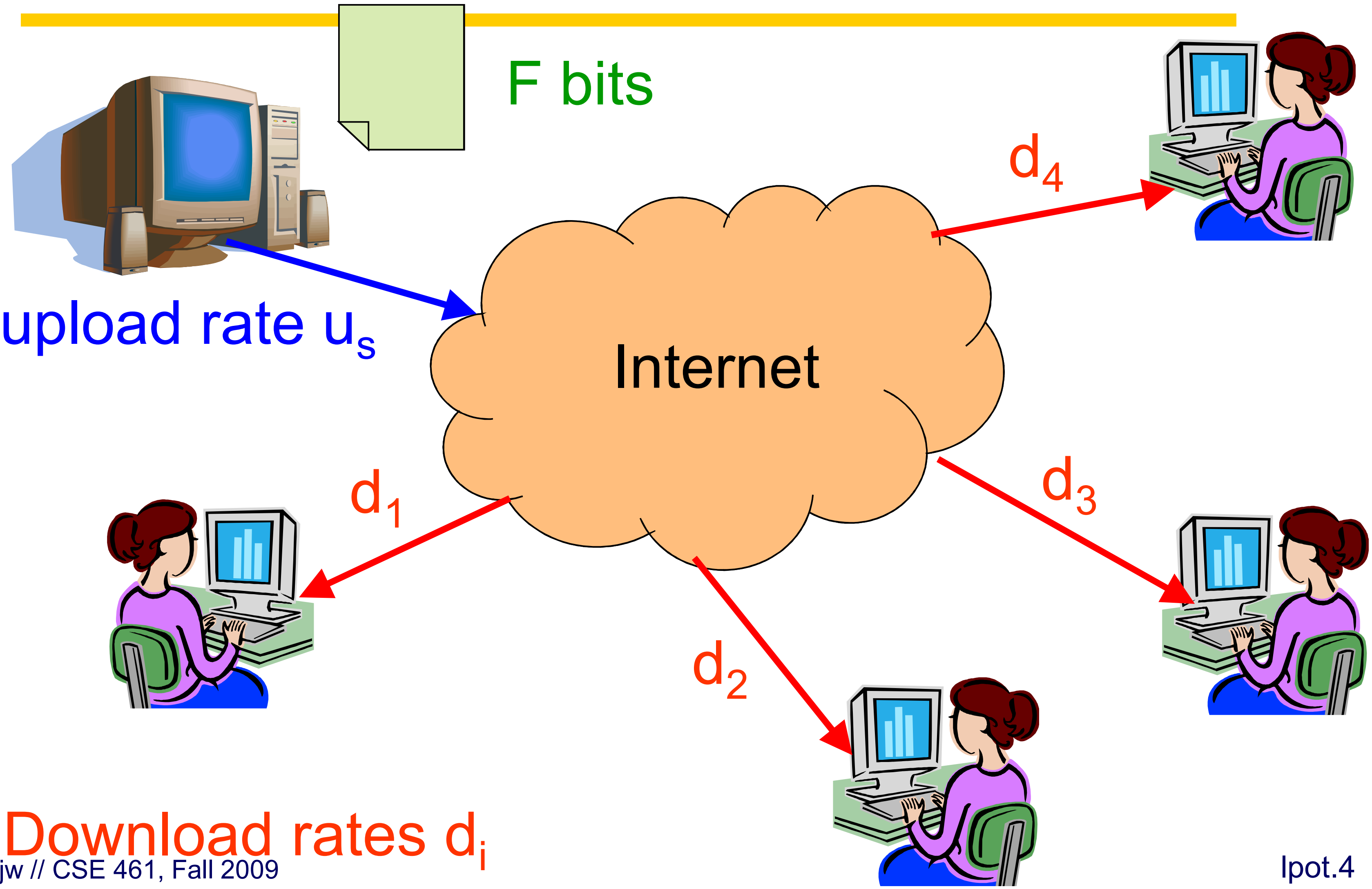
- P2P is simply the next iteration of scalable distributed systems

Client-Server Communication

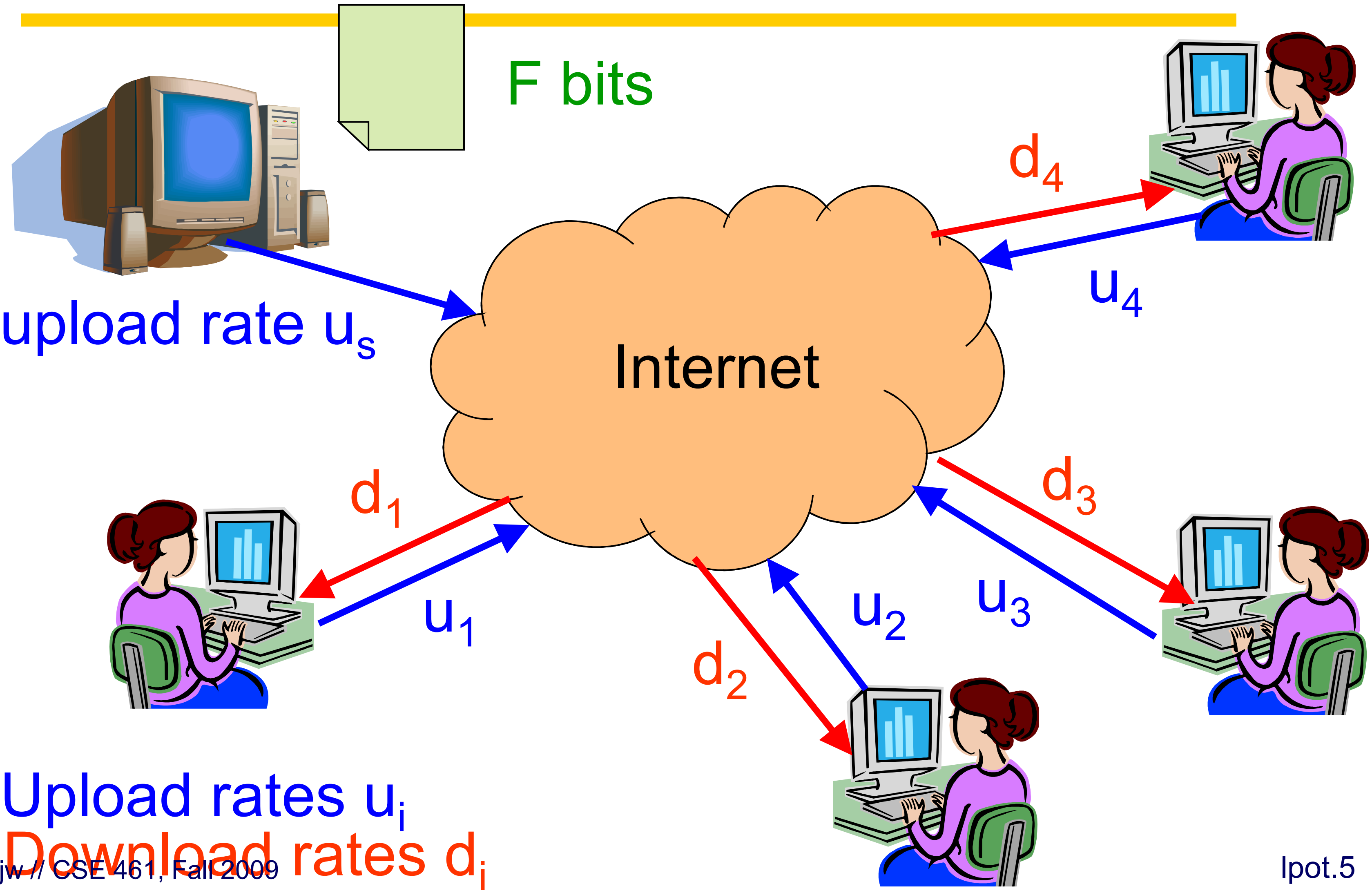
- Client “sometimes on”
  - Initiates a request to the server when interested
  - E.g., Web browser on your laptop or cell phone
  - Doesn’t communicate directly with other clients
  - Needs to know the server’s address
- Server is “always on”
  - Services requests from many client hosts
  - E.g., Web server for the www.cnn.com Web site
  - Doesn’t initiate contact with the clients
  - Needs a fixed, well-known address



Server Distributing a Large File



Peers Help Distributing a Large File



Upload rates u_i
Download rates d_i

Comparing the Two Models

- Peer-to-peer is self-scaling
  - Much lower demands on server bandwidth
  - Distribution time grows only slowly with N
- But…
  - Peers may come and go
  - Peers need to find each other
  - Peers need to be willing to help each other

Peer-to-Peer Networks: BitTorrent

- BitTorrent history and motivation
  - 2002: B. Cohen debuted BitTorrent
  - Key motivation: popular content
    - Popularity exhibits temporal locality (Flash Crowds)
    - E.g., Slashdot effect, CNN Web site on 9/11, release of movie/game
  - Focused on efficient *fetching*, not searching
    - Distribute same file to many peers
    - Single publisher, many downloaders
  - Preventing free-loading
- Significant fraction of Internet traffic today

BitTorrent: Tracker

- Infrastructure node
  - Keeps track of peers participating in the torrent
- Peers register with the tracker
  - Peer registers when it arrives
  - Peer periodically informs tracker it is still there
- Tracker selects peers for downloading
  - Returns a random set of peers
  - Including their IP addresses
  - So the new peer knows who to contact for data

BitTorrent: Peer and Chunks

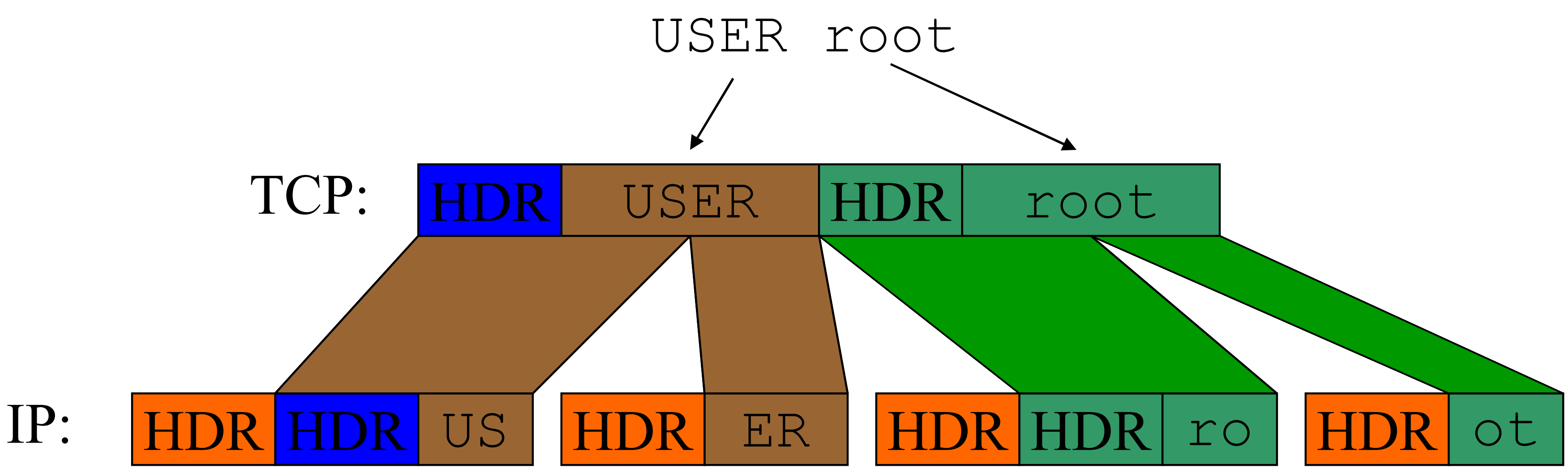
- Large file divided into smaller pieces
  - Fixed-sized chunks
  - Typical chunk size of 16KB - 256 KB
- Allows simultaneous transfers between peers
  - Downloading chunks from different neighbors
  - Uploading chunks to other neighbors
  - Favor neighbors that are contributing (incentives)
- Learning what chunks your neighbors have
  - Broadcast to neighbors when you have a chunk
- File done when all chunks are downloaded

Firewalls

- Originally, fairly basic: intent was to do per-packet inspection to block unused ports, for example
- Make sure we know exactly what's getting into the network and carefully think about their security
- Problem: a bug in your HTTP server (or its configuration) won't be caught by a basic firewall!
- Later firewalls became smarter – they'd reconstruct the flow. Keep per-flow state (previously impossible)
- Deny, for example, a HTTP request that contains "bobby tables".

Reconstructing Flows

- Let's say you want to search for the text "USER root". Is it enough to just search the data portion of TCP segments you see?



(Uh oh... we have to reassemble frags and resequence segs)

Fun with Fragments

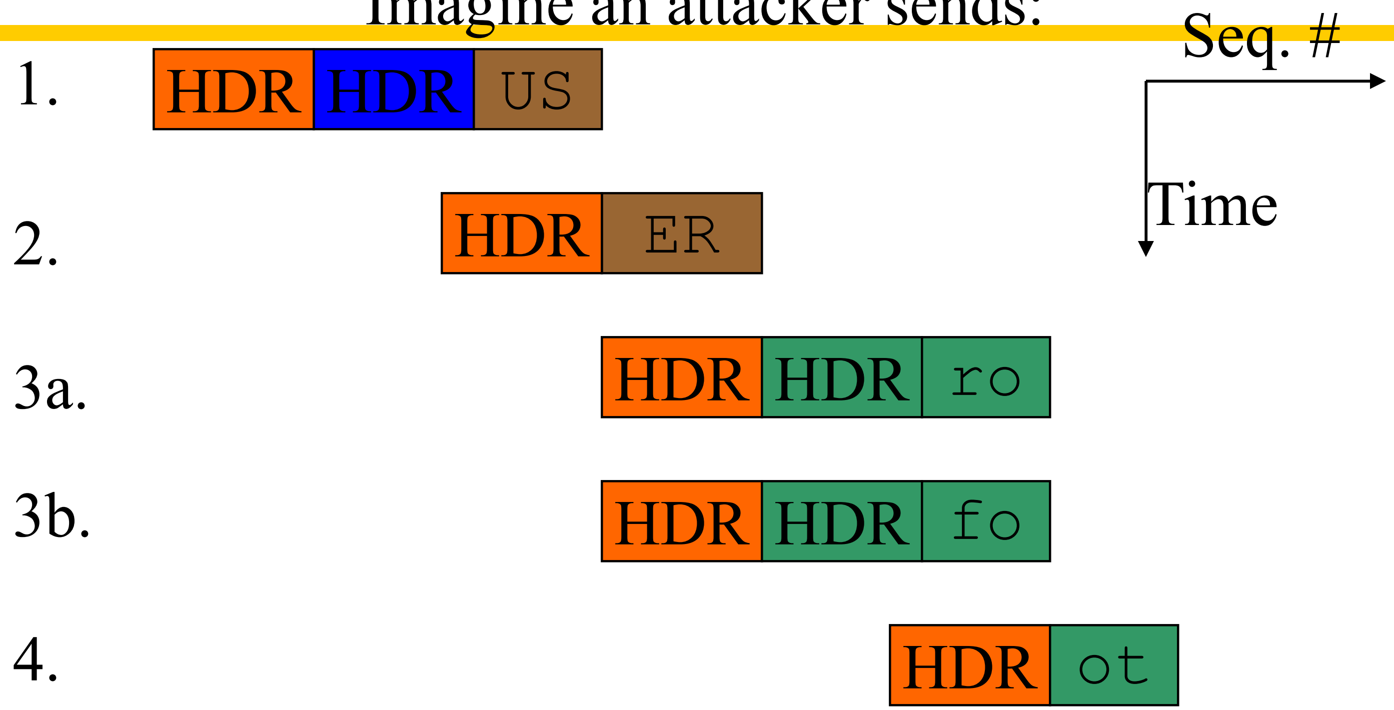
Imagine an attacker sends:

1. HDR HDR US
2. HDR ER
3. 1,000,000 unrelated fragments
4. HDR HDR ro
5. HDR ot

Think of the entire campus as being a massively parallel computer.
That supercomputer is solving the flow-reconstruction problem.
Now we're asking a single host to try to solve that same problem.

More Fragment Fun

Imagine an attacker sends:



Should we consider 3a part of the data stream "USER root"?
Or is 3b part of the data stream? "USER foot"!
- If the OS makes a different decision than the monitor: Bad.
- Even worse: Different OS's have different protocol interpretations, so it's impossible for a firewall to agree with all of them

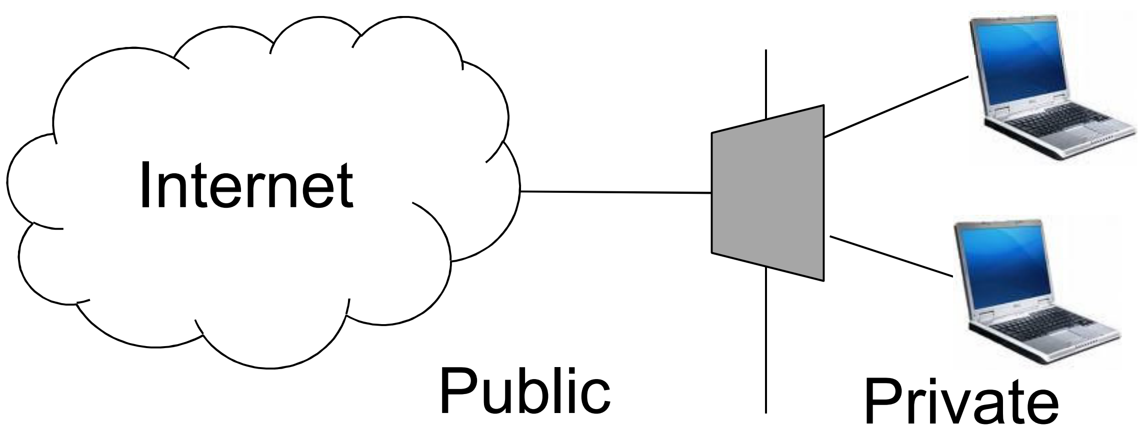
Trickery

- Non-standard parts of standards
  - IP fragment overlap behavior
  - TCP sequence number overlap behavior
  - Invalid combinations of TCP options
- Other ways to force a disparity between the monitor and the end-station
  - TTL
  - Checksum
  - Overflowing monitor buffers

See http://www.secnet.com/papers/ids-html/ for detailed examples

Network Address Translation (NAT)

- Originally motivated to multiplex multiple computers (e.g., at home) onto a single public IP (e.g., all your ISP gave you)
  – Now provides security/privacy too (often w/ firewalls etc.)



Private IPs and TCP/UDP ports mapped to public IP and port

– Mapping setup when you open a new connection, or configured

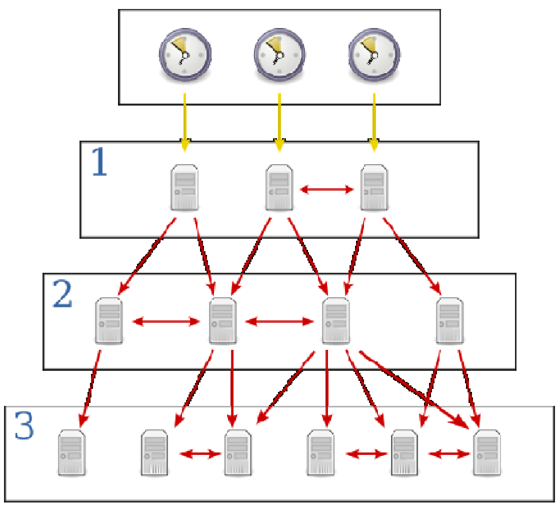
– Example of a middlebox (IP++ device in the network)

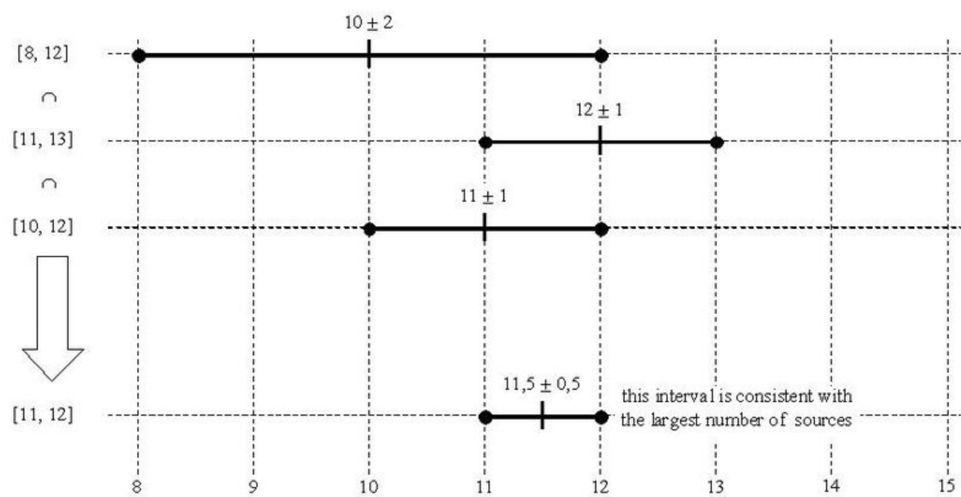
Network Time Protocol (NTP)

- Synchronizes clocks of Internet hosts to true time (UTC)
- Built on UDP, client exchanges timestamps with servers
- Timestamps processed (filtered) to estimate true time < 10ms
- Client clock slewed to track true time (avoid time jumps!)

NTP features

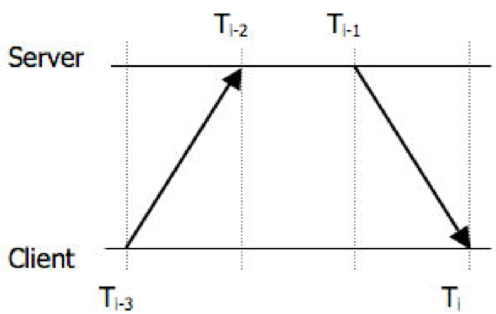
- Hierarchy of servers (stratums) to avoid dependency cycles
  - Atomic clock source at the top
- Look for agreement between multiple servers for sanity



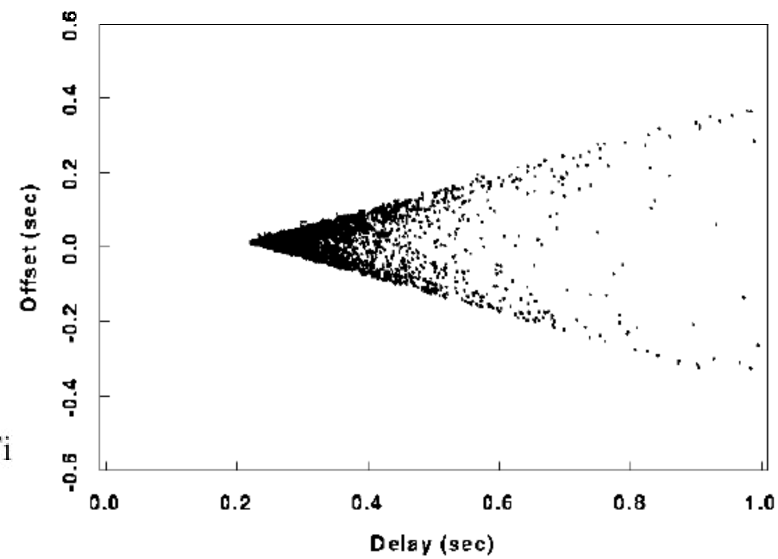


NTP timestamps

- Client/server exchange timestamps; assume symmetric delays
- Best estimates for lowest transit delays (no queuing)



$$a = T_{i-2} - T_{i-3}$$
$$b = T_{i-1} - T_i$$
$$\delta_i = a - b$$
$$\theta_i = \frac{a+b}{2}$$



Where:

θ_i: Clock offset of the server relative to the client at time Ti

δ_i: Round trip delay at time Ti

Virtual Private Networks (VPNs)

- Connect a private network via tunnels over the Internet
  - Private network is isolated; tunnels secured, e.g., with IPSEC