I apologize for being away last week...

Lecture 3: Application Protocols

Grading

- **Final grade will be computed as:**

$$(1-r)\left(\frac{\mathrm{exam}+\mathrm{lab}}{2}\right)+r\cdot\max(\mathrm{exam},\mathrm{lab})$$

- $r \sim 1/3$
- $r = \frac{1+q}{6}$
  - q is average grade on pop quizzes
  - Doing well on pop quizzes pushes grade closer to max
  - If you ace labs+exams, quizzes don’t matter

Pop Quizzes

- **5 minutes at beginning of class, open book**
- **Test on reading for the day: simple questions**
- **You can turn in the quiz within 12 hours of lecture if and only if**
  - You are an SCPD student
  - You are unable to make class due to a medical emergency or serious committment (e.g., college sports, wedding, funeral) and email us the day before
  - You have made a prior arrangement with instructors

Section Information

- Fridays, 11:00-11:50 AM, Skilling 193

Overview

- **End-to-end model**
- **Applications: power of reliable sockets**
  - Telnet
  - Web/HTTP
  - BitTorrent
  - Skype
- **Details of reliable transport**
- **Next lecture: TCP congestion control**

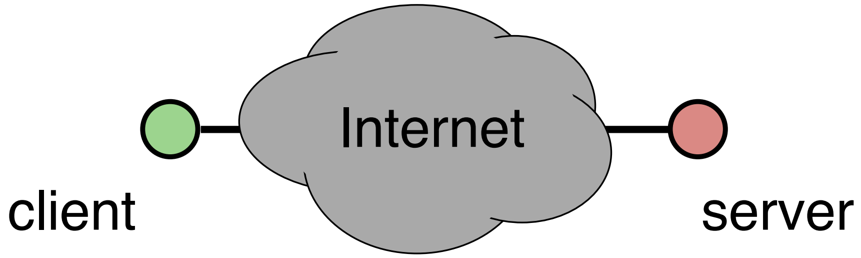
Review

- **The Internet is a network of networks**
- **Hourglass / narrow waist: host-host delivery**
  - IP protocol to deliver packets to hosts
  - Transport above for service/program demultiplexing
  - Link layer below for a single hop
- **Sockets are system call interface for communication**

Applications

- **Simple socket abstraction is very powerful**
  - Applications can use them in many ways
  - We'll go through four examples
- **"the intelligence is end to end rather than hidden in the network." (RFC 1958)**
- **Two basic questions for an application**
  - What do you send
  - Whom do you send it to

An Application View of the World



- **Why doesn’t the network help more?**
  - Compress data
  - Reformat/improve requests
  - Respond with cached data

End-to-End Model

- Saltzer, Reed, and Clark, 1984
- “The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.). We call this ... ’the end-to-end argument.”’
- Example: file transfer

File Transfer

- What can go wrong?
- How can we be sure a file arrives successfully?
- Why do something in the middle?

Strong End-to-End

- “The network’s job is to transmit datagrams as efficiently and flexibly as possible. Everything else should be done at the fringes..” (RFC 1958)
- Why?

Net Neutrality

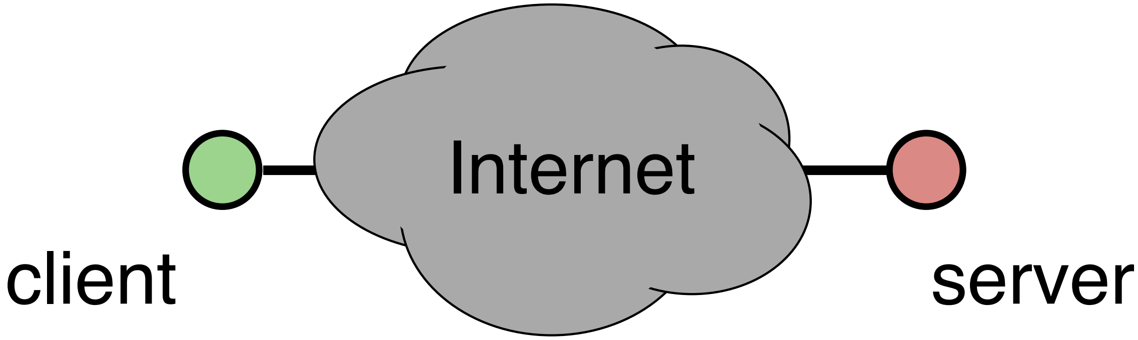
- Problem: Preferential treatment of some traffic over others
- Allows IP service providers to control applications
- Provider goals are not the same as end user's
- August 2007: Comcast forges RSTs to BitTorrent
- September 21, 2009: FCC issues 6 guidelines to support net neutrality

Performance definitions

- **Throughput – Number of bits/time you can transmit**
  - Improves with technology
- **Latency – How long for message to cross network**
  - Propagation + Transmit + Queue
  - We are stuck with speed of light…
    10s of milliseconds to cross country
- **Goodput – TransferSize/Latency**
- **Jitter – Variation in latency**
- **What matters most for your application?**

Telnet

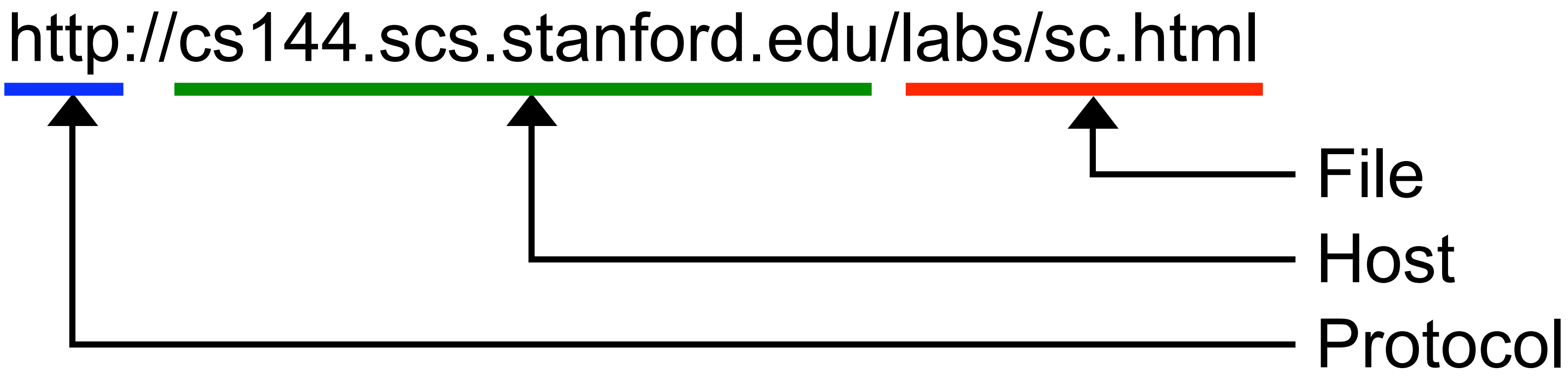
- **Server listens on port 23**
- **On connection, provides a login prompt**
- **Allows insecure remote logins**
- **Telnet client is a text interface**
  - Allows session management
  - Handles terminal control signals, etc.
- **What matters most?**



Web/HTTP

- Server lists on port 80
- On connection, waits for a request
- Protocol (but not data...) is in ASCII
- Sends response, maybe closes connection (client can ask it to stay open)

Parsing a URL



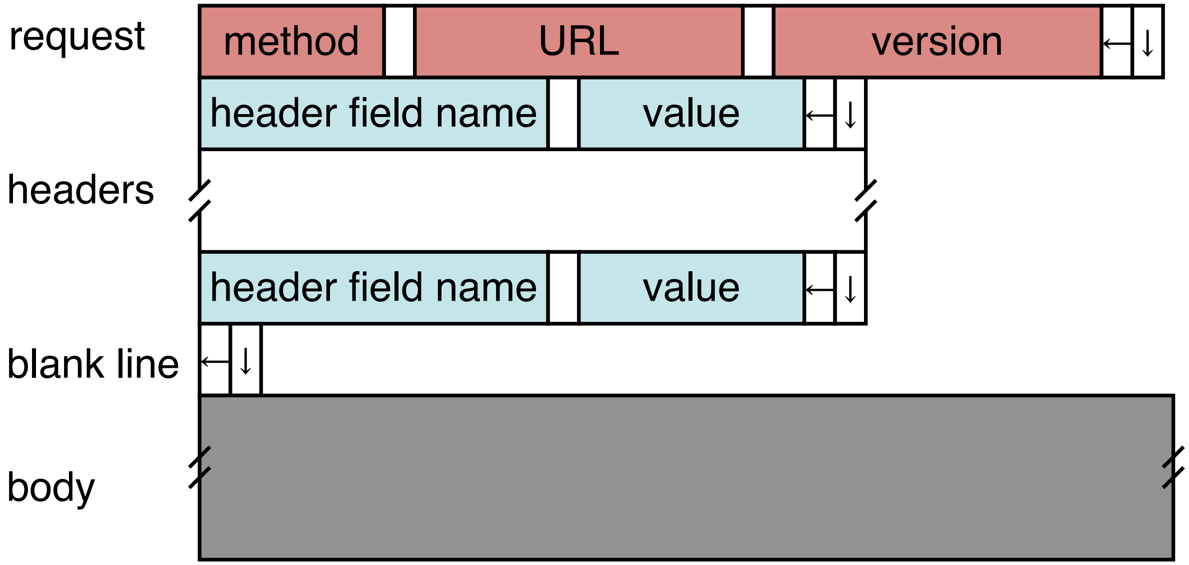
http://cs144.scs.stanford.edu/labs/sc.html

File

Host

Protocol

HTTP Request Format

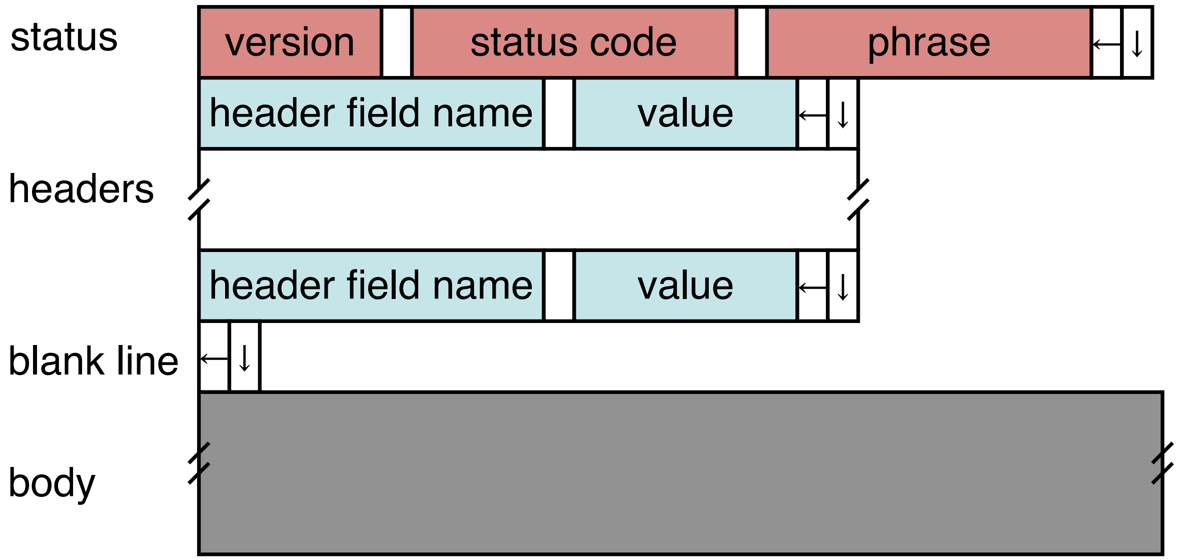


- **Request types: GET, PUT, POST, HEAD, PUT, DELETE**
- **A trivial request:** `GET / HTTP/1.0`
- **A browser request:** `http://localhost:8000`

A Browser Request

```
GET / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macinto ...
Accept: text/xml,application/xm ...
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

HTTP Response Format



- 1xx codes: Informational
- 2xx codes: Successes
- 3xx codes: Redirection
- 4xx codes: Client Error, 5xx codes: Server Error

www.scs.stanford.edu Response

```
HTTP/1.1 200 OK
Date: Thu, 03 Apr 2008 21:09:57 GMT
Server: Apache
Last-Modified: Thu, 03 Jan 2008 01:05:54 GMT
ETag: "a577678157a762e3d46afd4038d9a35bf2ae9386"
Accept-Ranges: bytes
Content-Length: 3586
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
...
```

HTTP Performance

- **What matters most?**
- **Different kinds of requests**
  - Lots of small requests (loading a page)
  - Big request (fetching a download)
- **Require different solutions**

Small Requests

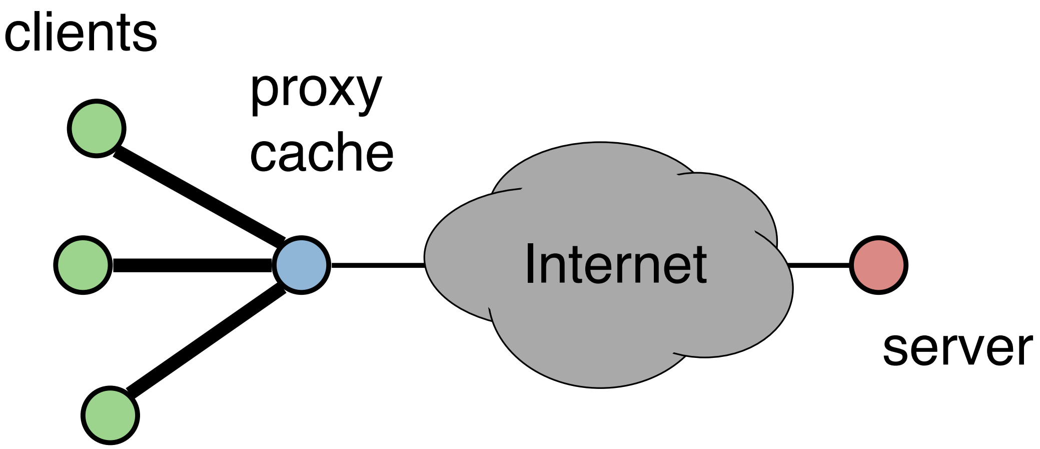
- **Latency matters**
- **Governed by round-trip-time (RTT) between hosts**
- **Two major causes:**
  - Opening a TCP connection
  - Data response-request
- **Solution: persistent connections**

Browser Request, Revisited

```
GET / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macinto ...
Accept: text/xml,application/xm ...
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

Big Requests

- **Problem is throughput on edge link**
- **Use an *HTTP proxy cache***
  - Can also improve latency!



Stale Caches

- **Items in the cache can go stale (you don't want to read yesterday's paper)**
- **Cache needs a way to conditionally ask for a document**
- **Issue a conditional GET (`If-modified-since` header)**
  - Server can reply with a `304 Not Modified`

```
GET / HTTP/1.1
Host: cs144.scs.stanford.edu
If-modified-since: Wed, 2 April 2008 08:00:00
```

Client-Server vs. Peer-to-Peer

- **Server can be a bottleneck**
  - Download time can scale $O(n)$ with n clients
  - Scaling up server bandwidth can be expensive (CDNs, lecture 8)
  - Slashdotting/flash crowds
- **Peer-to-peer: get a bunch of end-hosts to collaboratively distribute content**
- **A common peer-to-peer challenge is finding whom to collaborate with**

BitTorrent

- **Torrent file (`.torrent`) describes file to download**
  - Names *tracker*, URL that describes who is participating
  - File length, piece length, SHA1 hashes of pieces
  - Additional metadata (who created torrent, etc.)
- **Client contacts tracker, starts communicating with peers**

```
d8:announce47:http://vip.tracker.thepiratebay.org:80/announce7:comment20:
THANKS FOR SEEDING!10:created by13:uTorrent/177013:creation
datei1207365453e8:encoding5:UTF-84:infod6:lengthi839833600e4:
name18:XXX XXXXXXXXX.avi12:piece lengthi1048576e6:pieces16020:U ....
```

Pieces and Sub-pieces

- **BitTorrent breaks up a file into N pieces**
  - For throughput, pieces are large: 256KB-1MB
  - For latency, broken into subpieces
- **Hashes of pieces in torrent provide end-to-end integrity (HBO/Rome)**
  - Hashes computes a short summary of a piece of data
  - Cryptographically strong hashes: hard to find collisions or data that produces a hash (more in lectures 16+17)

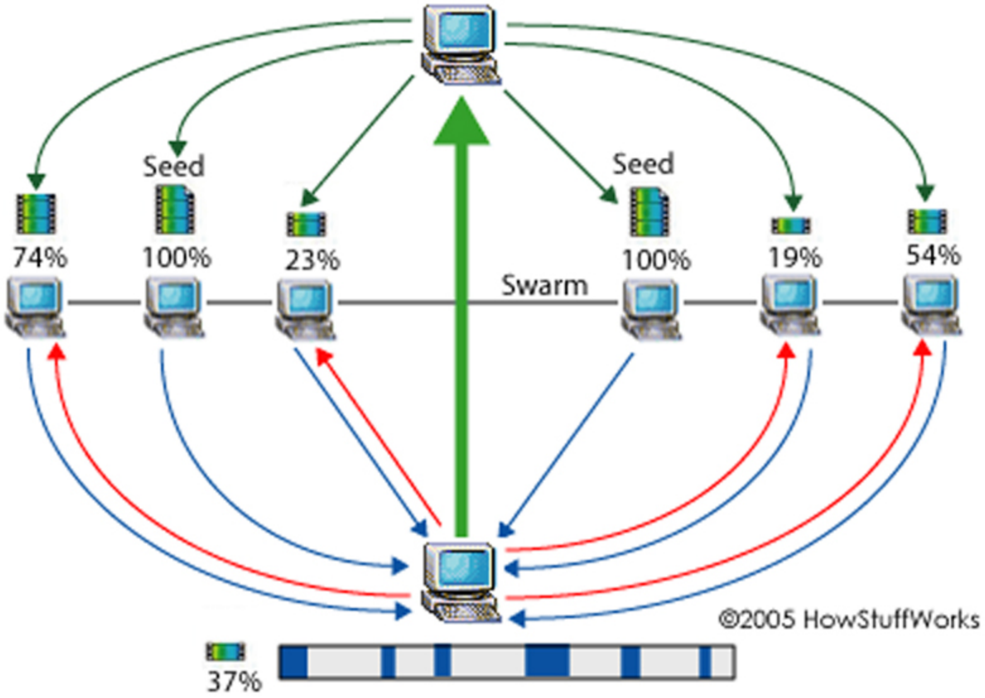
Whom To Talk To?

- **Uses a *Tit-for-Tat* (TFT) policy: upload only to those who give you data**
- **Most peers are “choked” and get no data**
- **Order unchoked peers by download rate, choke all but P best (e.g., 4, $\sqrt{C}$)**
- **Occasionally unchoke a random peer (might find its way into P best)**

What to Say?

- **Peers exchange metadata on what pieces they have**
- **Download rarest pieces**
- **When down to the last few pieces, ask for them from multiple peers**

BitTorrent Communication



BitTyrant

- **Optional reading: 2007 research paper**
- **Take advantage of altriusm**
  - Many peers give more than they take
  - Rate-limit yourself just enough to be unchoked
  - Connect to more peers instead of giving each peer more
- **Leads to a median 70% performance gain!**

BitTorrent Today: Research (SIGCOMM/NSDI/OSDI)

- Can a coordinator improve swarm performance by load-balancing peers?
- What do swarms look like in terms of communication patterns and size?
- How does one swarm streaming video?

BitTorrent Today: Engineering (IETF)

- Locality can improve performance (lower RTT)
- Hard for clients to know who might be local
- Can ISP collaborate to reduce transit costs and improve performance?
- Issues of contributory infringement...
- Application Layer Transport Optimization (alto)
- Low Extra Delay Background Transport (ledbat)

2-minute stretch

Skype

- **Real-time communication (voice, IM, etc.)**
- **Two major challenges**
  - Finding what host a user is on
  - Being able to communicate with arbitrary hosts
- **All Skype traffic is encrypted**
  - Researchers have mostly reverse-engineered the protocol
  - A few details are still unknown

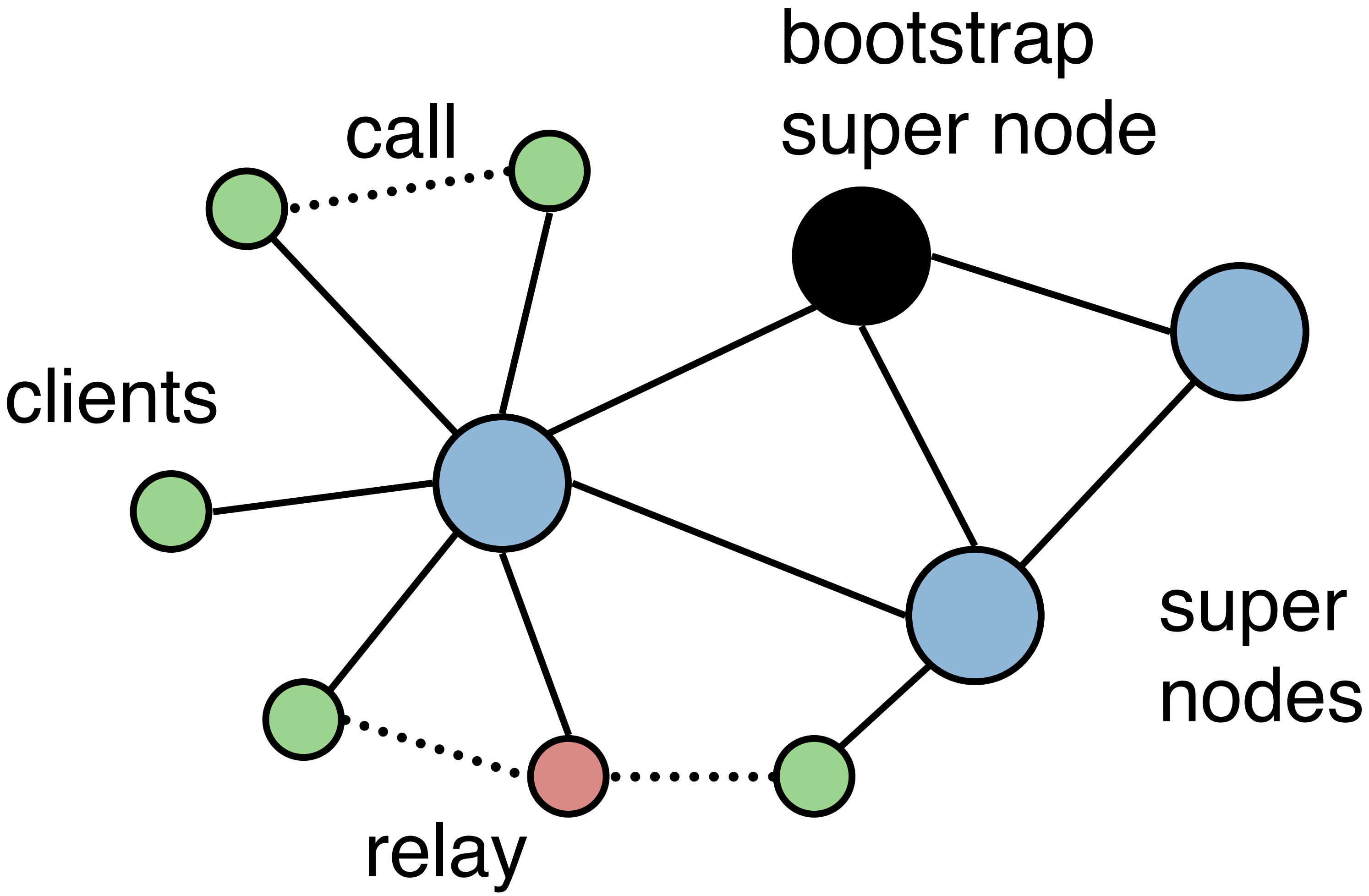
Finding a User

- **Distributed index across *super-peers***
  - Super-peers cannot be behind a NAT
  - There are 7 bootstrap super-peers: `66.235.180.9`, `66.235.181.9`, `212.72.49.143`, `195.215.8.145`, `64.246.49.60`, `64.246.49.61`, `64.246.48.24`
- **Uncertain exactly how this index is organized**
- **As number of peers grows, so does number of super-peers maintaining the index (or Skype, Inc. itself can add them)**

Talking to End-Hosts

- **What if neither host allows incoming connections?**
  - This is a common issue with NATs
  - Skype is more fragile to this that BitTorrent, because BitTorrent can connect to any peer, while Skype wants to connect to a specific peer
- **Skype uses *relays***

Skype Communication Architecture

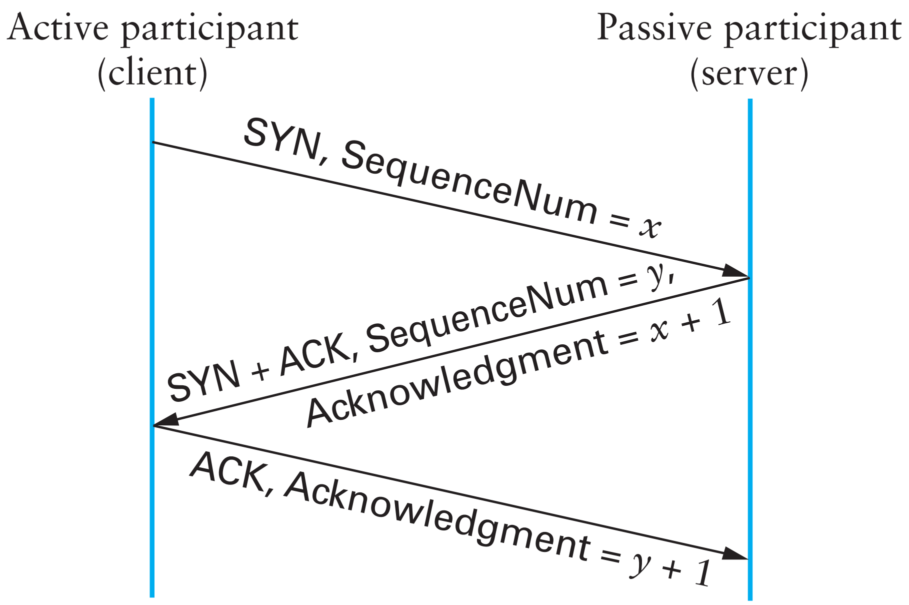


Skype Summary

- Uses a distributed index to find end-hosts
- Clients communicate directly, use relays when NATs are a problem

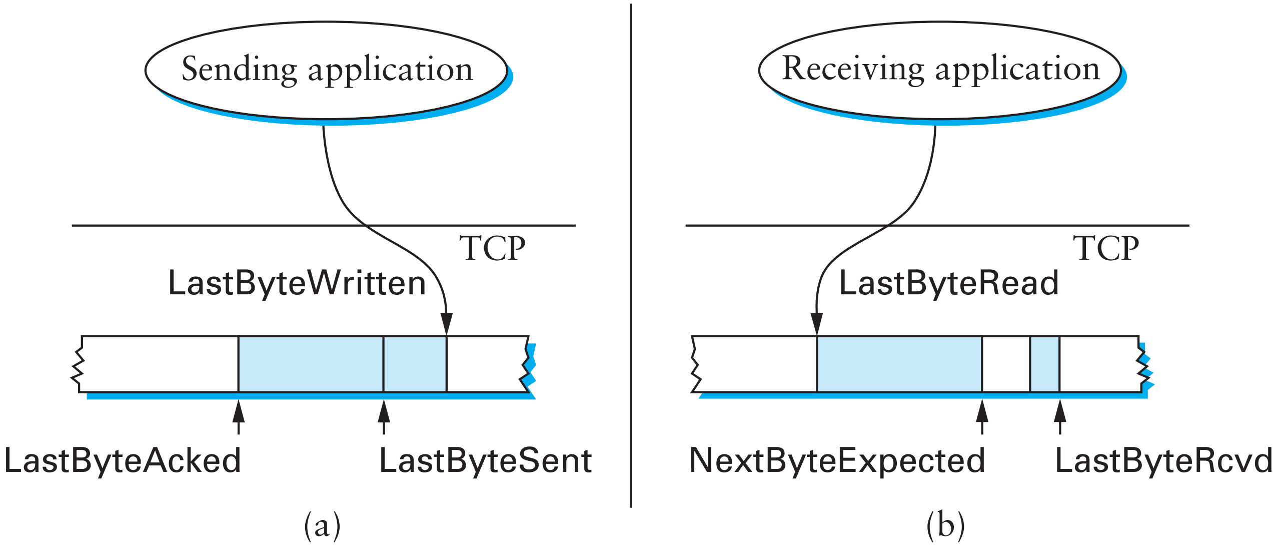
Details of Reliable Transport

Connection establishment



- **Need SYN packet in each direction**
  - Typically second SYN also acknowledges first
  - Supports "simultaneous open," seldom used in practice
- **If no program listening: server sends RST**
- **If server backlog exceeded: ignore SYN**
- **If no SYN-ACK received: retry, timeout**

Sliding window revisited



- **Used to guarantee reliable & in-order delivery**
- **New: Used for *flow control***
  - Instead of fixed window size, receiver sends AdvertisedWindow

Delayed ACKs

- **Goal: Piggy-back ACKs on data**
  - Echo server just echoes, why send separate ack first?
  - Delay ACKs for 200 msec in case application sends data
  - If more data received, immediately ACK second segment
  - Note: Never delay duplicate ACKs (if segment out of order)
- **Warning: Can interact *very* badly with Nagle**
  - "My login has 200 msec delays"
  - Set TCP_NODELAY

Retransmission

- TCP dynamically estimates round trip time
- If segment goes unacknowledged, must retransmit
- Use exponential backoff (in case loss from congestion)
- After $\sim$10 minutes, give up and reset connection
- Problem: Don't necessarily want to halt everything for one lost packet

Other details

- **Persist timer**
  - Sender can block because of 0-sized receive window
  - Receiver may open window, but ACK message lost
  - Sender keeps probing (sending one byte beyond window)
- **Keepalives**
  - Detect dead connection even when no data to send
  - E.g., remote login server, and client rebooted
  - Solution: Send “illegal” segments with no data
  - Remote side will RST (if rebooted), or timeout (if crashed)

32-bit seqno wrap around

Bandwidth	Time Until Wrap Around
T1 (1.5 Mbps)	6.4 hours
Ethernet (10 Mbps)	57 minutes
T3 (45 Mbps)	13 minutes
FDDI (100 Mbps)	6 minutes
STS-3 (155 Mbps)	4 minutes
STS-12 (622 Mbps)	55 seconds
STS-24 (1.2 Gbps)	28 seconds

Keeping the pipe full w. 100 msec delay

Bandwidth	Delay × Bandwidth Product
T1 (1.5 Mbps)	18KB
Ethernet (10 Mbps)	122KB
T3 (45 Mbps)	549KB
FDDI (100 Mbps)	1.2MB
STS-3 (155 Mbps)	1.8MB
STS-12 (622 Mbps)	7.4MB
STS-24 (1.2 Gbps)	14.8MB

TCP Extensions

- Implemented as header options
- Store timestamp in outgoing segments
- Extend sequence space with 32-bit timestamp (PAWS)
- Shift (scale) advertised window

Limitations of Flow Control

- *Link* **may be the bottleneck**
- **Sending too fast loses many packets**
- **Many retransmissions, lost acks, poor performance**
- **Flow control provides** *correctness*
- **Need more for performance: congestion control**

Overview

- **End-to-end model**
- **Applications: power of reliable sockets**
  - Telnet
  - Web/HTTP
  - BitTorrent
  - Skype
- **Details of reliable transport**
- **Next lecture: TCP congestion control**