Application Layer

In this Module

Application Layer

- Web and HTTP
- FTP
- Electronic mail: SMTP, POP3, IMAP
- DNS
- Principles of network applications
- P2P applications

Overview of Application Layer

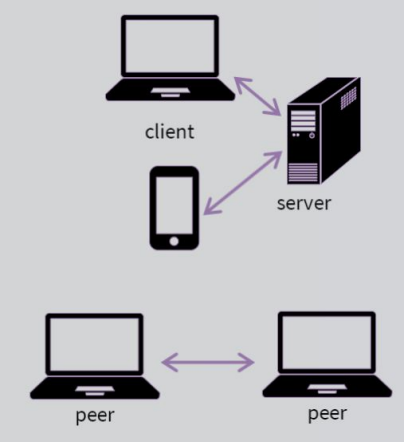
Our Goals:

- Conceptual, implementation aspects of network application protocols
    - Transport-layer service models
    - Client-server paradigm
    - Peer-to-peer paradigm
- Learn about protocols by examining popular application-level protocols
    - HTTP
    - FTP
    - SMTP/POP3/IMAP
    - DNS
- Creating networking applications
    - Socket API

Creating a Network App

- No need to write software for network-core devices
- Network-core devices do not run user applications
- Applications on end systems allows for rapid app development, propagation

Application Architectures



There are 2 types of application architectures:

- Client-Server
  - Server
    - Always-on host
    - Permanent IP address
    - Data centers for scaling
  - Client
    - Communicate with server
    - May be intermittently connected
    - May have dynamic IP addresses
    - Do not communicate directly with each other
- Peer-to-peer (P2P)
  - No always-on server
  - Arbitrary end systems directly communicate
  - Peers request service from other peers, provide service in return to other peers
  - Complex management peers are intermittently connected and change IP addresses
  - Self-scalability - new peers bring new service capacity, as well as new service demands

---

Process Communicating

- Client process
  Process that initiates communication
- Server process
  Process that waits to be contacted

Process

- Running within a host
- Within same host, two processes communicate using inter-process communication (defined by OS)
- Process in different hosts communicate by exchanging messages

---

Addressing Processes

- To receive messages, process must have identifier
- Host device has unique 32-bit IP address

- *Identifier* includes both IP address and port numbers associated with the process on host.
- Example port numbers:
    - HTTP server: 80
    - Mail server: 25
- To send HTTP message to gaia.cs.umass.edu web server:
    - IP address: 128.119.245.12
    - Port number: 80

App-Layer Protocol

Defines

- Types of messages exchanged:
    - E.g., request, response
- Message syntax
    - What fields in messages & how fields are delineated
- Message semantics
    - Meaning of information in fields
- Rules for when and how processes send and respond to messages

Open protocols:

- Defines in RFCs
- Allows for interoperability
- E.g., HTTP, SMTP

Proprietary protocols:

- E.g., Skype

Transport Service for an App

- Some apps (eg file transfer, web transactions) require 100% reliable data transfer
- Other apps (eg audio) can tolerate some loss
- Some apps (eg Internet telephony, interactive games) require low delay to be “effective”
- Some apps (eg multimedia) require min amount of throughput to be “effective”
- Other apps (“elastic apps”) make use of whatever throughput they get
- Encryption, data integrity…

Internet Transport Protocols Services

TCP service:

- Reliable transport between sending and receiving process
- *Flow control*: sender won't overwhelm receiver
- *Congestion control*: throttle sender when network overloaded
- *Does not provide*: timing, min throughput guarantee, security
- *Connection-oriented*: setup required between client and server processes.

UDP service:

- Unreliable data transfer between sending and receiving processes
- Does not provide: reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup.
- *40% less bandwidth requirement compared to TCP

Web and HTTP

- Objects: HTML file, JPEG image, Java applet, audio file,...
- Base HTML-file which includes several referenced objects
- Each object is addressable by a URL
  www.someschool.edu/dept/pic.gif
  Host name path name

HTTP Overview

HTTP is “stateless”

- Server maintains no information about past client requests

Uses TCP:

- Client initiatives TCP connection (creates socket) to server, port 80
- Server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP Connections

Non-persistent HTTP

- At most one object sent over TCP connection
  - Connection then closed
- Downloading multiple objects require multiple connections

Persistent HTTP

- Multiple object can be sent over single TCP connection between client, server

Non-Persistent HTTP

1a. Initiates TCP connection at www.someschool.edu on port 80 2. Client sends HTTP request messages into TCP connection socket. MEssage indicates the object dept/home.index 5. Receives response containing html file, displays html. Parsing html file, finds the referenced jpeg object.	1b. Waiting at port 80 to accept connection, notifying client 3. Receives request message, form response, send message into its socket. 4. Closes TCP connection

6. Steps 1-5 repeated for each of 10 jpeg objects

Non-Persistent HTTP: Response Time

RTT

- Time for a small packet to travel from client to server and back

HTTP response time = 2 RTT + file transmissions time

Persistent HTTP

Persistent HTTP:

- Server leaves connection open after sending response
- Subsequent HTTP messages between same client/server sent over open connection
- Client sends requests as soon as it encounters a referenced object
- As little as one RTT for all the referenced objects

Non-persistent HTTP issues:

- Requires 2 RTTs per object
- OS overhead for each TCP connection
- Browsers often open parallel TCP connections to fetch referenced objects

HTTP Response Status Codes

Appear in 1st line in server-to-client response message.

Some sample codes:

200 OK

- Request succeeded, requested object later in this msg

404 Not Found

- Requested document not found on this server

505 HTTP Version Not Supported

301 Moved Permanently

- Requested object moved, new location specified later in this msg

400 Bad Request

- Request msg not understood by server

User-Server State: Cookies

Example:

- Susan always accesses Internet from PC
- Visits specific e-commerce site for first time
- When initial HTTP requests arrives at site, site creates:
    - Unique ID
    - Entry in backend database for ID

Four components:

1) Cookie header line of HTTP response message
2) Cookie header line in next HTTP request message
3) Cookie file kept on user's host, managed by user's browser
4) Back-end database at Web site

Cookies

What cookies can be used for:

- Authorization
- Shopping carts
- Recommendations
- User session state (Web email)

Cookies and privacy:

- Cookies permit sites to learn a lot about you
- You may supply name and email to sites

Web Caches (Proxy Server)

Goal

Satisfy client request without involving origin server

- User sets browser: Web accesses via cache
- Browser sends all HTTP requests to cache
- Object in cache: cache returns object
- Else cache requests object from origin server, then returns object to client

- Cache acts as both client and server
    - Server for original requesting client
    - Client to origin server
- Typically cache is installed by ISP (university, company, residential ISP)
- Reduce response time for client request
- Reduce traffic on an institution's access link

Caching Example: Fatter Access Link

Assumptions:

- Avg object size: 100k bits
- Avg request rate from browsers to origin servers 15/sec
- Avg data rate to browsers: 1.50 Mbps
- RTT from institutional router to any origin server: 2 sec

Consequences:

- LAN utilization: 15%
- Access link utilization = 99%
- Total delay = Internet delay + access delay + LAN delay = 2 sec + mins + usecs

---

Conditional GET

- *Goal*: don't send object if cache has up to date cached version
    - No object transmission delay
    - Lower link utilization
- *Cache:* specify date of cached copy in HTTP request
  If-modified-since: <date>
- *Server:* response contains no object if cached copy is up to date:
  HTTP/1.0 304 Not Modified

FTP: The File Transfer Protocol

- Transfer file to/from remote host
- Client/server model
    - *Client:* side that initiates transfer (either to/from remote)
    - *Server*: remote host
- FTP: RFC 959
- FTP server: port 21

Electronic Mail

Roadmap

- Principles of network applications:
  - App architectures
  - App requirements
- Web and HTTP
- FTP
- Electronic mail: SMTP, POP3, IMAP
- DNS
- P2P applications

User agents

- AKA “mail reader”
- Composing editing, reading mail messages
- EG Outlook, Thunderbird, iPhone mail client

Mail Server

- Mailbox contains incoming messages for user
- Message queue of outgoing (to be sent) mail messages

Electronic Mail: STMP [RFC 2821]

- Uses TCP to reliably transfer email messages from client to server, port 25
- Direct transfer: sending server to receiving server
- Three phases of transfer
  - Handshaking (greeting)
  - Transfer of messages
  - Closure

Scenario: Alice Emails Bob

1. Alice composes a message
2. UA sends message to her mail server; massage place in message queue
3. Client side of SMTP opens TCP connection with Bob’s mail server
4. SMTP client side sends Alice’s message over the TCP connection
5. Bob’s mail server places the message in his mailbox
6. Bob reads message

Mail Access Protocols

SMTP:

Delivery/storage to receiver’s server

Mail access protocol:

- POP: Post Office Protocol [RFC 1939]:
  authorization, download
- IMAP: Internet Mail Access Protocol [RFC 1730]: more features
- HTTP: Gmail, Hotmail, Yahoo! Mail, etc.

POP3 and IMAP

More about POP3

- Previous example uses POP3 “download and delete” mode
- Bob cannot re-read email if he changes client
- POP3 “download-and-keep”: copies of messages on different clients
- POP3 is stateless across sessions
    - Ex: message on second PC says message is unread even though it was opened on the other PC

IMAP

- Keeps all messages in one place: at server
- Allows user to organize messages in folders
- Keeps suer state across sessions:
    - Names of folders and mappings between message IDs and folder name

---

DNS: Domain Name System

People: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- “Name”, e.g., www.yahoo.com - used by humans

Domain Name System:

- *Distributed database* implemented in hierarchy of many *name servers*
- *Application-layer protocol:* hosts, name servers communicate to *resolve* names (address/name translation)
    - Note: core Internet function, implemented as application-layer protocol
    - Complexity at network’s “edge”

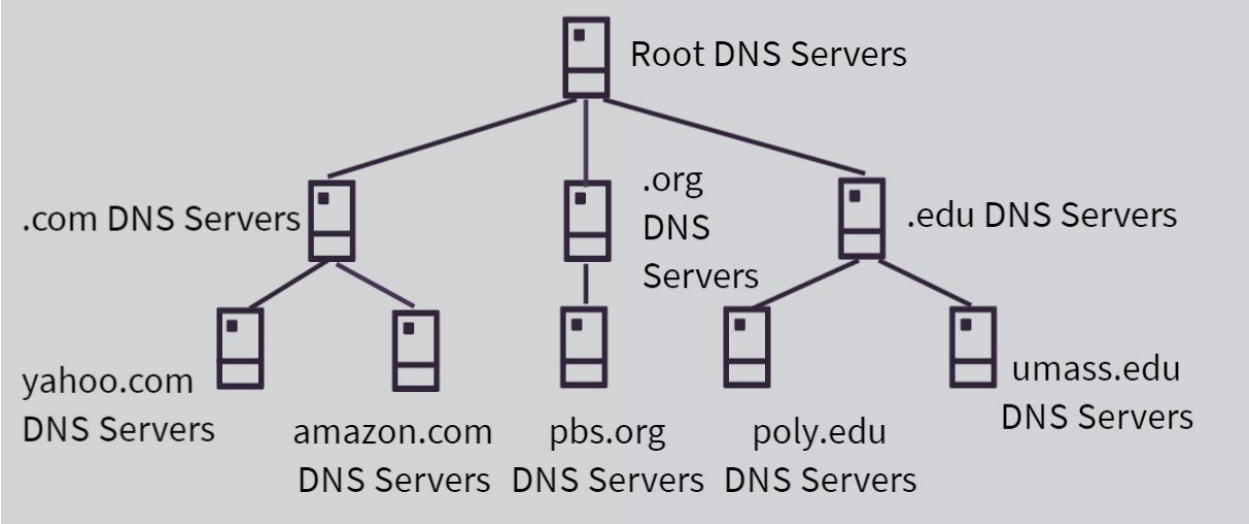
DNS: Services, Structure

DNS services

- Hostname to IP address translation

- Host aliasing
    - Canonical, alias names
- Mail server aliasing
- Load distribution
    - Replicated Web servers: many IP addresses correspond to one name
- *Why not centralize DNS?*
    - Single point of failure
    - Traffic volume
    - Distance centralized database
    - Maintenance
    - It doesn't scale

DNS: A Distributed, Hierarchical Database



Client wants UP for www.amazon.com; 1st approx:

- Client queries root server to find com DNS server
- Client queries .com DNS server to get amazon.com DNS server
- Client queries amazon.com DNS server to get IP address for www.amazon.com

---

DNS: Root Name Servers

- There are 13 root name servers worldwide
- Contacted by local name server that can not resolve name
- Root name server:
    - Contacts authoritative name server if name mapping not known
    - Gets mapping

- Returns the mapping to local name server

TLD, Authoritative Server

Top-level Domain (TLD) Servers:

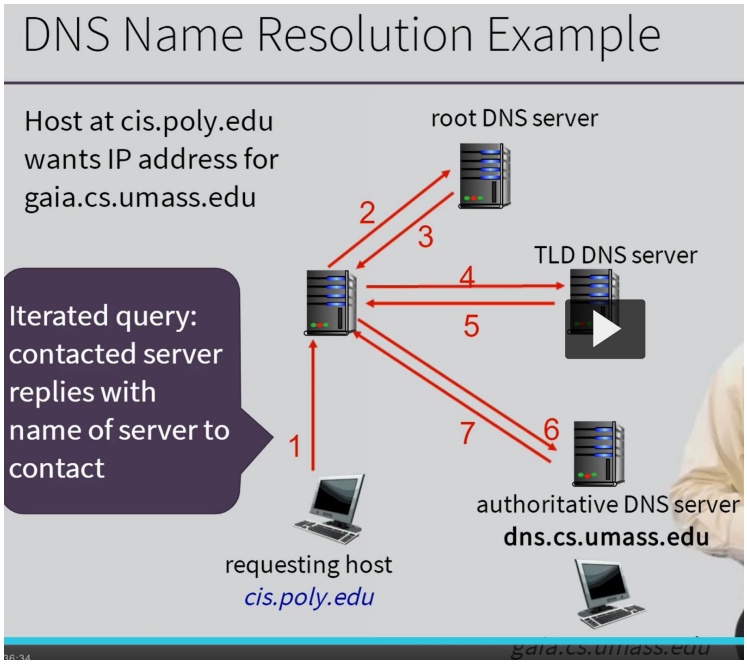
- Responsible for com, org, net, edu, aero, jobs, museums, and all top-level country domains, e.g: uk, fr, ca, jp
- Network Solutions maintains servers for .com
- Educause for .edu TLD

Authoritative DNS Servers:

- Organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named shots
- Can be maintain by organization or server provider

Local DNS Name Server

- Does not strictly belong to hierarchy
- Each ISP (residential ISP, company, university) has one
  - Also called "default name server"
- When host makes DNS query, query is sent to its local DNS server
  - Has a local cache of recent name-to-address translation pairs (but may be out of date!)
  - Acts as proxy, forwards query into hierarchy



DNS: Caching, Updating Records

- Once (any) name server learns mapping, it *caches* mapping
    - Cache entries timeout (disappear) after some time (TTL)
    - TLD servers typically cached in local name servers
        - Thus root name servers not often visited
    - Cached entries may be *out of date* (best effort name-to-address translation!)
        - If name host changes IP address, may not be known Internet-wide until all TTLs expire
    - Update/notify mechanisms proposed IETF standard
        - RFC 2136

DNS Records

DNS: distributed db storing resource records (RR)

RR format: (name, value, type, ttl)

type=A

- **Name** is hostname
- **Value** is IP address

type=NS

- **Name** is domain (eg foo.com)
- **Value** is hostname of authoritative name server for this domain

type=CNAME

- **Name** is alias name for some “canonical” (the real) name
- **www.ibm.com** is really **servereast.backup2.ibm.com**
- **Value** is canonical name

type=MX

- **Value** is name of mail server associated with **name**

Inserting Records into DNS

- Example: new startup “Network Utopia”
- Register name networkutopia.com at DNS *registrar* (eg Network Solutions)
    - Provide names, IP addresses of authoritative name servers (primary and secondary)
    - Registrar inserts two RRs into .com TLD server:
      (networkutopia.com,
      Dns1.networkutopia.com, NS)
      (dns1.networkutopia.com, 212.212.212.1, A)

- Create authoritative server type A record for www.networkutopica.com; type MX record for networkutopia.com

---

Attacking DNS

DDoS attacks

- Bombard root servers with traffic
    - Not successful to date
    - Traffic Filtering
    - Local DNS servers cache IPs of TLD servers, allowing root server bypass
- Bombard TLD servers
    - Potentially more dangerous

Redirect attacks

- Man-in-middle
    - Intercept queries
- DNS poisoning
    - Send bogus request to DNS server, which caches

Exploit DNS for DDoS

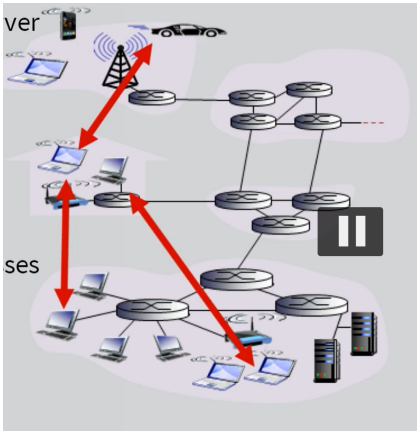
- Send queries with spoofed source address: target IP
- Requires amplification

P2P Applications

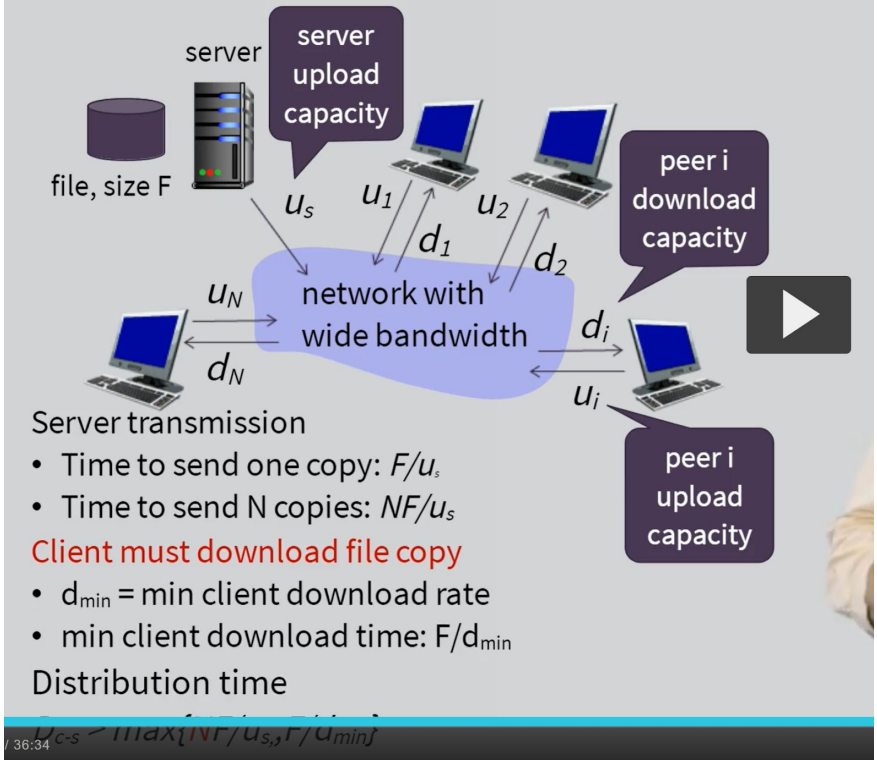
- *No* always-on server
- Arbitrary end systems directly communicate
- Peers are intermittently connected and change IP addresses

Examples:

- File distribution (BitTorrent)
- Streaming (KanKan)
- VoIP (Skype)



File Distribution: Client-Server



Server transmission

- Time to send one copy: F/u_s
- Time to send N copies: NF/u_s

Client must download file copy

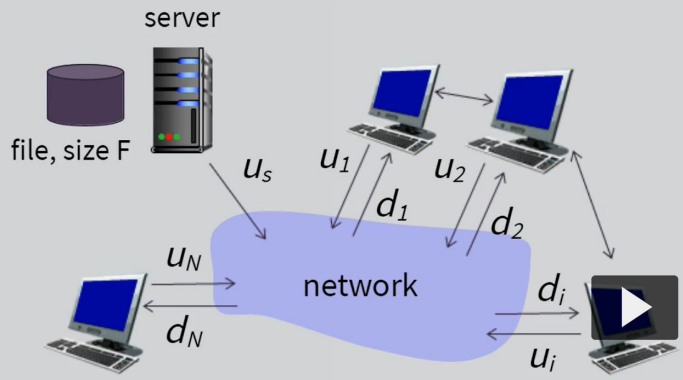
- d_{min} = min client download rate
- min client download time: F/d_{min}

Distribution time

$D_{c-s} > max\{NF/u_s, F/d_{min}\}$

36:34

File Distribution Time: P2P



Clients: as aggregate must download *NF* bits

- Max upload rate (limting max download rate) is $u_s + \Sigma u_i$

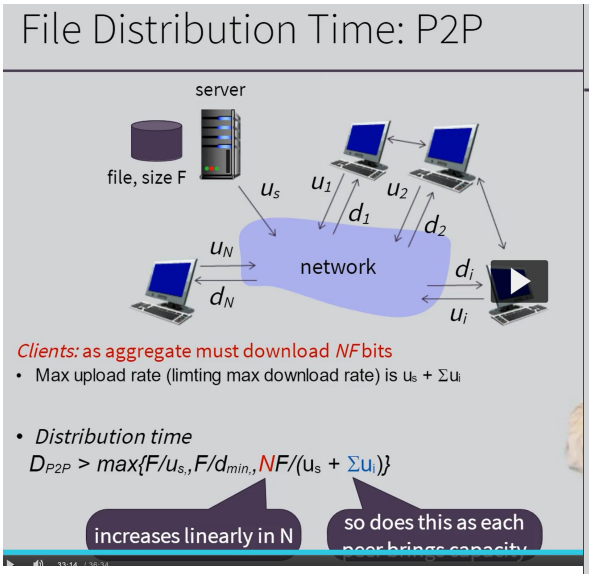
- *Distribution time*

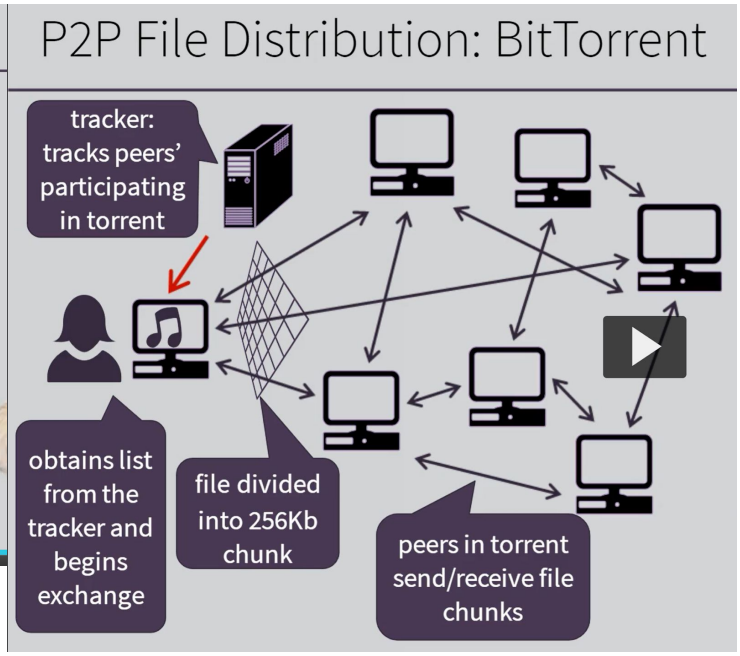
$D_{P2P} > max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$

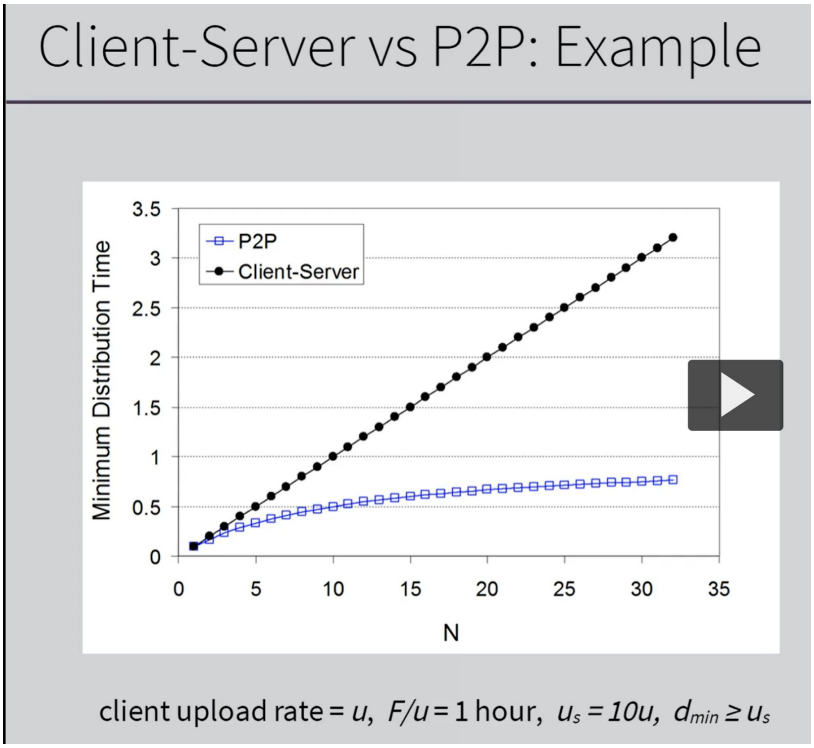
increases linearly in N

so does this as each peer brings capacity

33:14 / 36:34







Peering joining torrent:

- Has no chucks, but will accumulate them over time from other peers
- Registers with tracker to get list of peers, connects to subset of peers ("neighbors")

- Once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent

---

Distributing Hash Table (DHT)

- DHT: *a distributed P2P database*
- Database has (key, value) pairs; examples:
    - Key: ss number; value: human name
    - Key: movie title; value: IP address
- Distributed the (key, value) pairs over the (millions of peers)
- A peer queries DHT with key
    - DHT returns values that match the key

Summary

- Application architectures
    - Client-server
    - P2P
- Application service requirements:
    - Reliability, bandwidth, delay
- Internet transport service model
    - Connection-oriented, reliable: TCP
    - Unreliable, datagrams: UDP
- Specific protocols:
    - HTTP
    - FTP
    - SMTP, POP, IMAP
    - DNS
    - P2P: BitTorrent, DHT