Application Layer

Dr. A Krishna Chaitanya,
Indian Institute of Information Technology Sri City

30-Aug-2017

Network Applications

- Applications use the services of network (Transport layer)
- For an application developer, architecture and services of network are fixed
- Architectures of applications:
  - Client-Server architecture
  - Peer-to-Peer (P2P) architecture
- Application developer decides on the architecture and services of transport layer to be used.

Client-Server Architecture

- Server: An end system that serves the requests from various hosts.
- A server is always ON.
- Client: An end system that requests a server for content.
- A client can be either ON-OFF or always ON.
- Example applications using this architecture: web, e-mail, file transfer, etc.

Peer-to-Peer Architecture

- End systems communicate by a direct connection.
- The end systems are called peers.
- Example applications: skype, internet telephony, torrents, etc
- Advantages:
  - File distribution
  - Self-scalable: can handle growth in traffic
  - Cost effective: no server infrastructure and server bandwidth.
- Challenges in P2P Architecture:
  - ISP friendly: asymmetric data traffic.
  - Security
  - Incentives: Peers should share bandwidth.

Processes Communicating

- A process is a program that is running within an end system.
- A client process is a process running on a client and a server process is process running on a server.
- It is the client process and server processes that are actually communicating.
- A process sends and receives messages to and from transport layer through a software interface known as socket.
- A socket is also known as Application Programming Interface (API).

Services of Transport Layer

- Reliable data transfer: Guaranteed data delivery service.
- Throughput
- Timing: for example, it is guaranteed that a packet will be delivered no more than 100 msec later.
- security: end-point authentication, encryption and decryption.

Transport protocols

- Transmission Control Protocol (TCP)
  - Connection oriented service: handshaking, full-duplex connection
  - Reliable data transfer service: packets get delivered without error and in proper order.
  - Congestion control
- User Datagram Protocol (UDP)
  - Connectionless
  - Unreliable data transfer service.
  - No congestion control

Applications

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP [RFC 5321]	TCP
Remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
File transfer	FTP [RFC 959]	TCP
Streaming multimedia	HTTP (e.g., YouTube)	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary (e.g., Skype)	UDP or TCP

Addressing Processes

- There are many processes running on a host, how to identify the destination process?

Addressing Processes

- There are many processes running on a host, how to identify the destination process?
- We identify host by IP address.

Addressing Processes

- There are many processes running on a host, how to identify the destination process?
- We identify host by IP address.
- We identify processes by port numbers!
- For example, web server is identified by port number 80, mail server is identified by port number 25.

Web and HTTP

- A web page is a document and consists of objects
- An object is nothing but a file such as HyperText Markup Language (HTML) file, an image file, applet or video clip.
- If a web page contains a basic html file and ten images, we say the web page contains 11 objects.
- HyperText Transfer Protocol (HTTP) is the web's application layer protocol
- HTTP uses client-server architecture with TCP.
- The client program and server program talk to each other by exchanging HTTP messages.

Uniform Resource Locater

- An object should be addressable by a URL.
- Each URL consists of hostname and objects path name
- For example, http://www.iiits.ac.in/wp-content/uploads/2017/05/Untitled-design-15.png is url for an image.
- www.iiits.ac.in is host name
- wp-content/uploads/2017/05/Untitled-design-15.png is path name.
- Client side of HTTP is implemented in Web browser and server side is implemented in Web server.
- Examples: Apache and Microsoft Internet Information server.

HTTP

- HTTP client initiates a connection with HTTP server (handshaking).
- Once the connection is established, client and server exchange messages through socket interface.
- Client sends an HTTP request and receives HTTP messages through its socket
- Server receives HTTP requests and sends HTTP responses through its socket interface.
- Client/server need not worry about packets (does not have any control) after sending through their socket.
- Server sends requested files without storing state information of client. Thus HTTP is a stateless protocol.

HTTP Connection

- Let us say, a web page has one html file and 10 images.
- How does client retrieve the web page?

HTTP Connection

- Let us say, a web page has one html file and 10 images.
- How does client retrieve the web page?
- Nonpersistent and Persistent

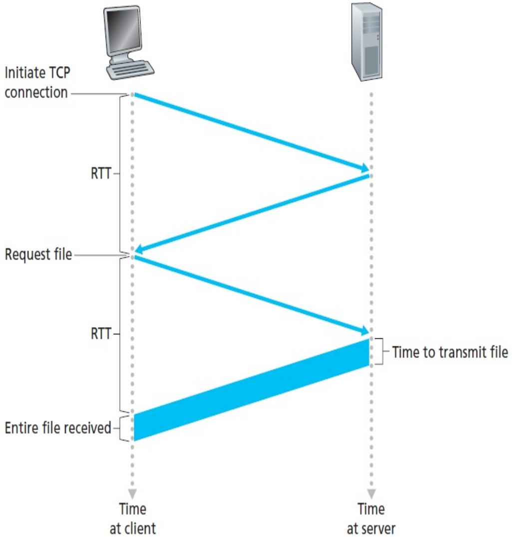
HTTP Connection

- Let us say, a web page has one html file and 10 images.
- How does client retrieve the web page?
- Nonpersistent and Persistent
- Nonpersistent: one TCP connection for each file
- Persistent: one TCP connection for all files

Nonpersistent Connection

- For each file:
  - HTTP client initiates a TCP connection to the server on port number 80
  - Client sends its HTTP request and it includes the path name to the file
  - HTTP server receives the request and retrieves the file and sends the HTTP response to the client
  - HTTP server tells TCP to close the connection.
- TCP connections can be serial or parallel depending on browser's configuration

Round-Trip Time



Persistent Connection

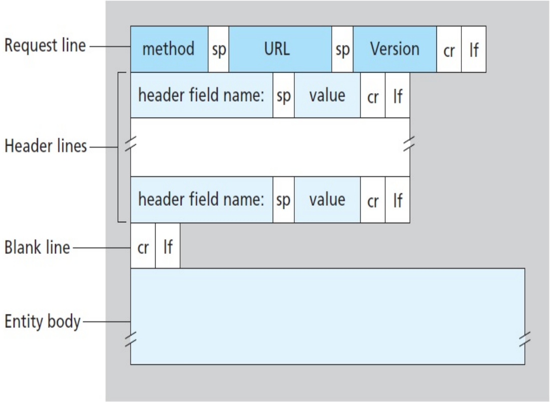
- Server leaves the connection after sending the HTTP response
- Pipelining: A browser can request for files without waiting for the reception of pending requests.
- TCP closes after some idle period
- Default mode HTTP: Persistent connection with pipelining.

HTTP Request Format

- HTTP request message:

  GET /somedir/page.html HTTP/1.1
  Host: www.iitm.ac.in
  Connection: close
  User-agent: Mozilla/4.0
  Accept-language: En
- Methods: GET, PUT, POST, HEAD, DELETE

HTTP Request

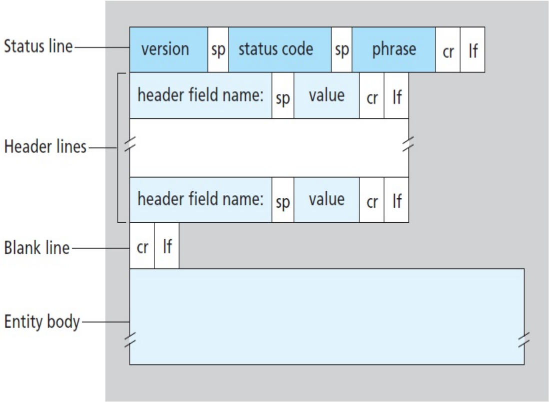


HTTP Response

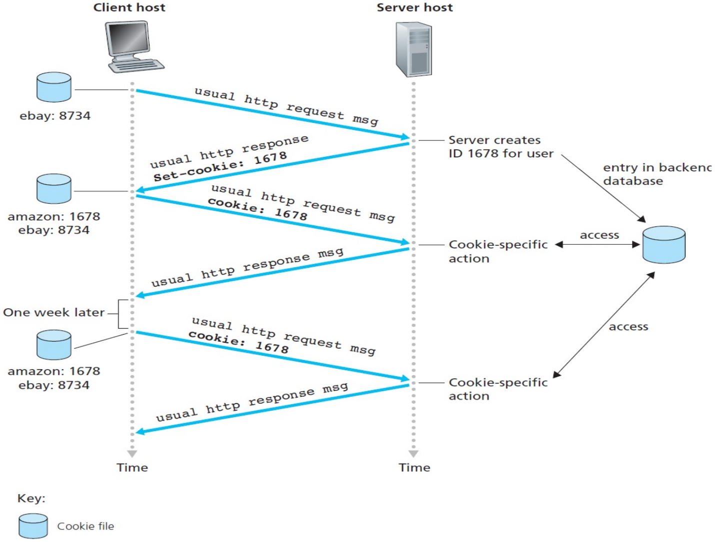
- HTTP response message:

  HTTP/1.1 200 OK
  Connection: close
  Date: Sat, 07 Jul 2007 12:00:15 GMT
  Server: Apache/1.3.0 (Unix)
  Last-Modified: Sun, 6 May 2007 09:23:24 GMT
  Content-length: 6821
  Content-Type: text/html
  (data data ... data)
- 200 OK
- 301 Moved Permanently
- 404 Not Found

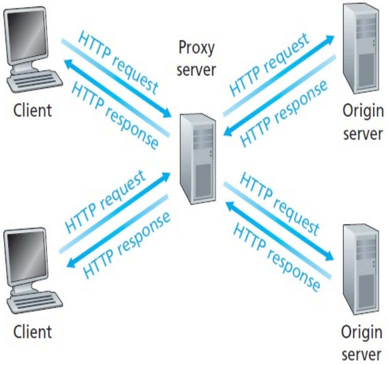
HTTP Response



Cookies


Client host
Server host
ebay: 8734
usual http request msg
Server creates
ID 1678 for user
usual http response
Set-cookie: 1678
entry in backenc
database
amazon: 1678
ebay: 8734
usual http request msg
cookie: 1678
Cookie-specific
action
access
usual http response msg
One week later
amazon: 1678
ebay: 8734
usual http request msg
cookie: 1678
access
Cookie-specific
action
usual http response msg
Time
Time
Key:
Cookie file

Web Caching



Client
HTTP request
HTTP response
Proxy server
HTTP request
HTTP response
Origin server
Client
HTTP request
HTTP response
HTTP request
HTTP response
Origin server

Conditional GET

- Cache request message:

  GET /somedir/student.jpg HTTP/1.1
  Host: www.iiits.ac.in
  If-modified-since: Wed, 23 Aug 2017 17:30:00

Conditional GET

- Cache request message:

  GET /somedir/student.jpg HTTP/1.1
  Host: www.iiits.ac.in
  If-modified-since: Wed, 23 Aug 2017 17:30:00

- Response: HTTP/1.1 304 Not Modified

  Date: Tue, 29 Aug 2017 13:00:00
  Server: Apache/1.3.0 (Unix)
  (empty empty empty)

File Transfer Protocol

- Similar to HTTP: client-server architecture, transmission control protocol
- Two parallel TCP connections to transfer a file: TCP control connection and TCP data connection
- Control information:
  - User identification
  - Change remote directory
  - Commands to put and get files
- FTP is said to control information out-of-band where as HTTP is said to control information in-band.

FTP

- Commands:
  - USER username
  - PASS password
  - LIST
  - RETR filename
  - STOR filename
- Replies:
  - 331 username OK, password required
  - 125 data connection already open; transfer starting
  - 425 can not open data connection
  - 452 error writing file

Electronic Mail

- Asynchronous communication medium
- Major components of e-mail system:
  - User agent: allows users to read, forward, save and compose messages
  - Mail server
  - SMTP
- Examples of user agents: Microsoft Outlook, Mozilla Thunderbird, Apple Mail

SMTP

- User agent sends message to user's mail server.
- SMTP transfers message from user's mail server to recipient's mail server.
- Client side of SMTP is running on sender's mail server and server side of SMTP is running on recipient's mail server.
- Recipient's mail server delivers the message in recipient's mail box.

SMTP Sequence of Operations

- Alice composes message using her user agent. Provides Bob's mail address and instructs to send the message.
- User agent sends the message to her mail server and message waits in the queue of the server.
- SMTP client sees the message in the mail server and it opens a TCP connection to an SMTP server running on Bob's mail server.
- SMTP transfers the message from client to server.
- SMTP server receives the message. Bob's mail server places the message in Bob's mail box.
- Bob invokes his user agent to read the message.

SMTP Sequence of Operations

- If recipient's mail server is down, SMTP client reattempts to send the message (say for every 30 minutes)
- If the delivery is not successful after some duration, it will be notified to the sender and message will be dropped.

Client-Server Conversation

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr ... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Message Formats

- Header lines similar to those in HTTP messages
- Header must have From:, To:
- Optional header lines include Subject:

Comparison with HTTP

- HTTP is a pull protocol
- SMTP is push protocol
- SMTP requires each message to be 7-bit ASCII format. HTTP does not have this restriction
- HTTP encapsulates each object in its own HTTP response message. Internet mail places all of its objects into one message.

Mail Access Protocols

- In early days of internet, Bob reads mail by logging onto mail server and executing a mail reader on that host
- Client-server architecture
- Reads e-mail by running a client on the user's end system
- Mail access protocol transfers message from Bob's mail server to his local PC.
- Popular mail access protocols: Post Office Protocol - version 3 (POP3), Internet Mail Access Protocol (IMAP) and HTTP

POP3

- Begins when a user agent opens a TCP connection with mail server on port 110.
- POP3 progresses in three phases:
  - Authorization
  - Transaction
  - Update
- Authorization: user <username> and pass <password>
- Transaction: user agent sends commands and server responds with +OK and -ERR

POP3 Transaction

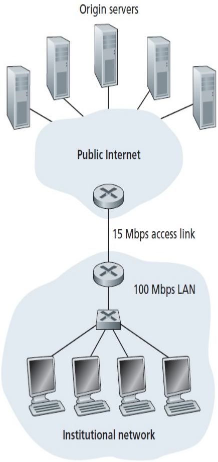
- Two modes:
  - download and delete
  - download and keep
- Download and delete:

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: (blah blah ...
S: .................
S: ..........blah)
S: .
C: dele 1
C: retr 2
S: (blah blah ...
S: .................
S: ..........blah)
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

IMAP and HTTP

- IMAP associates each message with a folder
- Provides commands to allow users to create folder and move messages across folders
- Provides commands to search for a message
- Maintains user state information across IMAP sessions
- Components of messages can be retrieved
- HTTP:
  - e-mail access through web browser
  - web browser communicates to the mail server via HTTP

Problem



- Average object size is 1Mbits
- Average request rate: 15 objects per sec.
- Time from request to receive response is 2 sec on an average.

Total Response Time

Total Response Time

- Traffic intensity on the LAN

Total Response Time

- Traffic intensity on the LAN
- 0.15

Total Response Time

- Traffic intensity on the LAN
- 0.15
- Traffic intensity on the access link

Total Response Time

- Traffic intensity on the LAN
- 0.15
- Traffic intensity on the access link
- 1

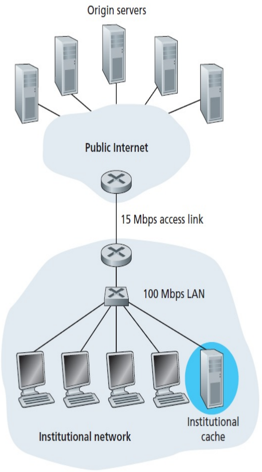
Total Response Time

- Traffic intensity on the LAN
- 0.15
- Traffic intensity on the access link
- 1
- Suppose, access link upgraded to 100Mbps link.

Total Response Time

- Traffic intensity on the LAN
- 0.15
- Traffic intensity on the access link
- 1
- Suppose, access link upgraded to 100Mbps link.
- Expensive Solution!

Problem



- Assume cache provides hit rate of 0.4

Total Response Time

- Traffic intensity on the LAN

Total Response Time

- Traffic intensity on the LAN
- 0.15

Total Response Time

- Traffic intensity on the LAN
- 0.15
- Traffic intensity on the access link

Total Response Time

- Traffic intensity on the LAN
- 0.15
- Traffic intensity on the access link
- 0.6

Total Response Time

- Traffic intensity on the LAN
- 0.15
- Traffic intensity on the access link
- 0.6
- Typically delay ten milliseconds

Total Response Time

- Traffic intensity on the LAN
- 0.15
- Traffic intensity on the access link
- 0.6
- Typically delay ten milliseconds
- Average response time: 0.4*(0.01)+0.6*(2.01) seconds

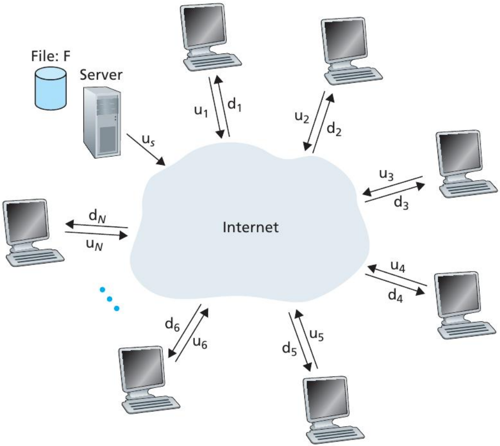
Applications of Peer-to-Peer Architecture

- File distribution: application that transfers a file from a single source to multiple peers.
- Database distributed over a large community of peers.
- Internet telephony : Skype.

File Distribution

- Each peer can redistribute any portion of the file to any other peer
- Popular file distribution protocol : BitTorrent,developed by Bram Cohen
- Scalability

Scalability



- N peers
- Distribution time: the time required to distribute a file to all peers.

Assumptions

- Internet has abundant bandwidth and all bottlenecks are in the network access
- All the server and client bandwidth is available for file distribution

Distribution Time for Client-Server Architecture

- Let D_{cs} denote the distribution time for client-server architecture for a file size of F bits
- The server has to transmit a total of NF bits at an upload rate of $u_s bps$.

Distribution Time for Client-Server Architecture

- Let D_{cs} denote the distribution time for client-server architecture for a file size of F bits
- The server has to transmit a total of NF bits at an upload rate of $u_s bps$.
- Minimum time required for distribution is $\frac{NF}{u_s}$ seconds

Distribution Time for Client-Server Architecture

- Let D_{cs} denote the distribution time for client-server architecture for a file size of F bits
- The server has to transmit a total of NF bits at an upload rate of $u_s bps$.
- Minimum time required for distribution is $\frac{NF}{u_s}$ seconds
- Let $d_{min} = \min\{d_1, \ldots, d_N\}$

Distribution Time for Client-Server Architecture

- Let D_{cs} denote the distribution time for client-server architecture for a file size of F bits
- The server has to transmit a total of NF bits at an upload rate of $u_s bps$.
- Minimum time required for distribution is $\frac{NF}{u_s}$ seconds
- Let $d_{min} = \min\{d_1, \ldots, d_N\}$
- Minimum distribution time is $\frac{F}{d_{min}}$ seconds

Distribution Time for Client-Server Architecture

- Let D_{cs} denote the distribution time for client-server architecture for a file size of F bits
- The server has to transmit a total of NF bits at an upload rate of $u_s bps$.
- Minimum time required for distribution is $\frac{NF}{u_s}$ seconds
- Let $d_{min} = \min\{d_1, \ldots, d_N\}$
- Minimum distribution time is $\frac{F}{d_{min}}$ seconds
- Thus,

$$D_{cs} \geq \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$

Distribution Time for Client-Server Architecture

- Let D_{cs} denote the distribution time for client-server architecture for a file size of F bits
- The server has to transmit a total of NF bits at an upload rate of $u_s bps$.
- Minimum time required for distribution is $\frac{NF}{u_s}$ seconds
- Let $d_{min} = \min\{d_1, \ldots, d_N\}$
- Minimum distribution time is $\frac{F}{d_{min}}$ seconds
- Thus,

$$D_{cs} \geq \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$

- Show that

$$D_{cs} = \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$

Distribution Time for P2P Architecture

- The server has to send each bit of the file at least once: Minimum distribution time is at least $\frac{F}{u_s}$ seconds

Distribution Time for P2P Architecture

- The server has to send each bit of the file at least once: Minimum distribution time is at least $\frac{F}{u_s}$ seconds
- The peer with lowest download rate can not obtain F bits in less than $\frac{F}{d_{min}}$ seconds

Distribution Time for P2P Architecture

- The server has to send each bit of the file at least once: Minimum distribution time is at least $\frac{F}{u_s}$ seconds
- The peer with lowest download rate can not obtain F bits in less than $\frac{F}{d_{min}}$ seconds
- The total upload rate $u_{total} = u_s + u_1 + \cdots + u_N$. The system must deliver F bits to each of the N peers: Minimum distribution time is $\frac{NF}{u_{total}}$

Distribution Time for P2P Architecture

- The server has to send each bit of the file at least once: Minimum distribution time is at least $\frac{F}{u_s}$ seconds
- The peer with lowest download rate can not obtain F bits in less than $\frac{F}{d_{min}}$ seconds
- The total upload rate $u_{total} = u_s + u_1 + \cdots + u_N$. The system must deliver F bits to each of the N peers: Minimum distribution time is $\frac{NF}{u_{total}}$
- Thus, minimum distribution time D_{P2P} is at least

$$\max\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_{total}}\}$$

Distribution Time for P2P Architecture

- The server has to send each bit of the file at least once: Minimum distribution time is at least $\frac{F}{u_s}$ seconds
- The peer with lowest download rate can not obtain F bits in less than $\frac{F}{d_{min}}$ seconds
- The total upload rate $u_{total} = u_s + u_1 + \cdots + u_N$. The system must deliver F bits to each of the N peers: Minimum distribution time is $\frac{NF}{u_{total}}$
- Thus, minimum distribution time D_{P2P} is at least

$$\max\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_{total}}\}$$

- Assumption: each peer can redistribute a bit as soon as it receives the bit.

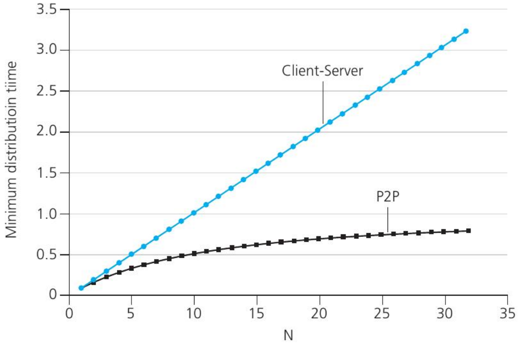
Distribution Time for P2P Architecture

- The server has to send each bit of the file at least once: Minimum distribution time is at least $\frac{F}{u_s}$ seconds
- The peer with lowest download rate can not obtain F bits in less than $\frac{F}{d_{min}}$ seconds
- The total upload rate $u_{total} = u_s + u_1 + \cdots + u_N$. The system must deliver F bits to each of the N peers: Minimum distribution time is $\frac{NF}{u_{total}}$
- Thus, minimum distribution time D_{P2P} is at least

$$\max\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_{total}}\}$$

- Assumption: each peer can redistribute a bit as soon as it receives the bit.
- There is a scheme that actually achieves this lower bound.

Distribution Time for P2P Architecture



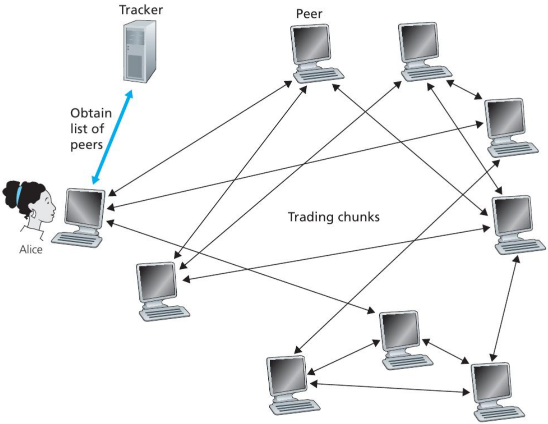
- All peers upload at a rate of u bps.
- $\frac{F}{u} = 1$ hour, $u_s = 10u$ and $d_{min} \geq u_s$.

Bit Torrent

- Collection of peers participating in the distribution of a file is called a torrent
- Peers in a torrent download equal-size chunks of the file (typically 256 KBytes)
- A peer accumulates more and more chunks over time
- Once a peer has acquired complete file, it may leave the torrent or continue to participate in the torrent
- Peers may leave torrents with subsets of chunks

Bit Torrent

- Each torrent has a node called tracker.
- When a peer joins the torrent, it registers with the tracker
- Each peer in the torrent periodically updates the tracker about its presence.



Bit Torrent

- Alice receives a subset of participating peers in the torrent
- She establishes TCP connection with some of the peers and we call them as neighboring peers of Alice
- Neighboring peers may vary over time
- Each peer will have some subset of chunks from the file, with different peers having different subsets
- Alice maintains a list of chunks that her neighbors have.

- Alice will issue requests for chunks she currently does not have
- Which chunks should be requested first?

Bit Torrent

- Alice will issue requests for chunks she currently does not have
- To which of her neighbors should she send requested chunks?

Bit Torrent

- Alice will issue requests for chunks she currently does not have
- Which chunks should be requested first?

- Rarest first: finds the chunks that are rarest among her neighbors

Bit Torrent

- Alice will issue requests for chunks she currently does not have
- To which of her neighbors should she send requested chunks?
- Tit-for-tat

Tit-for-tat

- Alice gives priority to the neighbors that are currently supplying her data at the highest rate
- Every 30 seconds, she also picks one additional neighbor at random and sends it chunks. Let it be Bob.
- In due course of time, Alice, may become one of the top uploaders in which case Bob could start sending data to Alice.

Distributed Hash Tables (DHT)

- Huge database to be stored among number of peers in a distributed way
- Database is consists of (key, value) pairs. For Example, (PAN No., Aadhar No.), (Content Name, IP), etc.
- Peers query the database by supplying the key and database replies the matching pairs to the querying peer
- How to store database among the peers

DHT

- Assign an identifier to each peer.
- An identifier is an integer in $[0, 2^n - 1]$ for some fixed n
- (key, value) pairs are also identified by integers using hash functions
- Hash function is available to all peers.

Storing in DHT

- Define a rule for assigning keys to peers

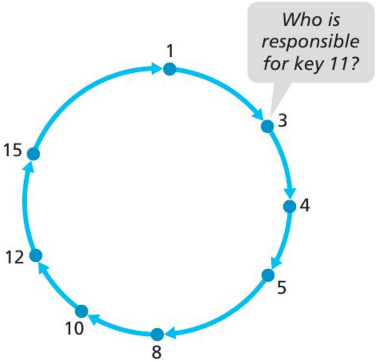
Storing in DHT

- Define a rule for assigning keys to peers
- Closest to the key:
- For example, $n = 4$, with eight peers: 1,3,4,5,8,10,12 and 15. Store (11, 0123-4567-8910) in one of the eight peers

Storing in DHT

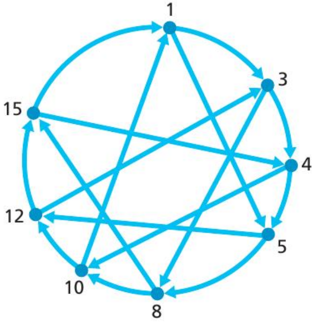
- Define a rule for assigning keys to peers
- Closest to the key:
- For example, $n = 4$, with eight peers: 1,3,4,5,8,10,12 and 15. Store (11, 0123-4567-8910) in one of the eight peers
- By closest convention, peer 12 is the immediate successor for key 11. Store in peer 12.
- If the key is larger than all the peer identifiers, we use modulo-2^n convention.

Circular DHT



- Each peer is aware of only its immediate predecessor and successor
- N messages at most

Shortcut



- Number of shortcuts are relatively small in number
- How many shortcut neighbors and which peers should be these shortcut neighbors? Research problem: $O(\log(N))$

Peer Churn

- Peers can come and go without warning
- Peers keep track to two immediate predecessor and successors.
- When a peer abruptly leaves, its predecessor and successor learn that a peer has left and updates the list of its predecessor and successor.