Root NS

(startup.com, dns.startup.com, NS, -)

(dns.startup.com, 192.50.100.21, A, -)

(www.startup.com, 192.50.100.8, A, -)

(startup.com, www.startup.com, CNAME, -)

authoritative NS

(foo.startup.com, 192.50.100.10, A, -)

(startup.com, mail.startup.com, MX, -)

(mail.startup.com, 192.50.100.5, A, -)

Matta @ BUCS - Applications 1-64

64

Inserting records into DNS

- Example: just created startup "Network Utopia"
- Register name networkutopia.com at a registrar (e.g., Network Solutions, delegated by ICANN)
  - Need to provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
  - Registrar inserts two RRs into the com top-level server:

```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```

- Put in authoritative server Type A record for www.networkuptopia.com and Type MX record for networkutopia.com

Matta @ BUCS - Applications 1-65

65

DNS protocol, messages

DNS protocol : *query* and *reply* messages over UDP, both with same *message format*

msg header

- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative

identification	flags
number of questions	number of answer RRs
number of authority RRs	number of additional RRs
questions (variable number of questions)	
answers (variable number of resource records)	
authority (variable number of resource records)	
additional information (variable number of resource records)	

12 bytes

Matta @ BUCS - Applications 1-66

66

DNS protocol, messages

cs.bu.edu, MX

Name, type fields for a query

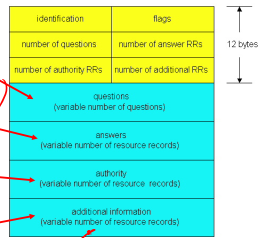
(cs.bu.edu, mail.cs.bu.edu, MX, -)

RRs in reponse to query

records of authoritative servers (records of type "NS")

additional "helpful" info that may be used (e.g., IP address of mail server)

(mail.cs.bu.edu, 192.10.5.1, A, -)



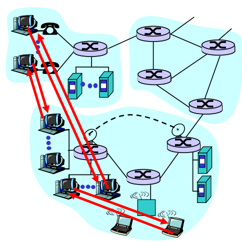
67

Beyond client-server: P2P architecture

- no always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses
- examples: BitTorrent (Vuze client), Skype

All peers are servers = highly scalable!

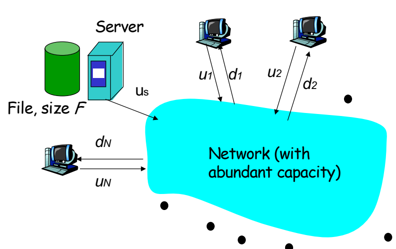
But difficult to manage



68

File Distribution: Server-Client vs P2P

Question : How much time to distribute file from one server to *N peers*?



u_s: server upload capacity

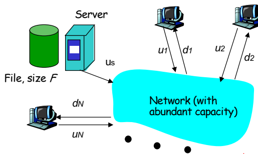
u_i: peer i upload capacity

d_i: peer i download capacity

69

File Distribution: Server-Client vs P2P

Question : How much time to distribute file from one server to *N peers*?



bottleneck C_1 C_2

Throughput $= \min(c_1, c_2)$

Delay $\geq \dfrac{F}{\min(c_1, c_2)}$

$\max\left(\dfrac{F}{c_1}, \dfrac{F}{c_2}\right)$

$D_{c/s} \geq \max\left(\dfrac{NF}{u_s}, \dfrac{F}{d_{min}}\right)$

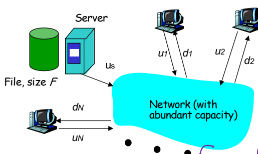
$= \dfrac{NF}{u_s}$

$(d_{min} \gg u_s/N)$

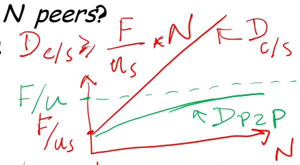
70

File Distribution: Server-Client vs P2P

Question : How much time to distribute file from one server to *N peers*?



$D_{c/s} \geq \dfrac{F}{u_s} \times N$



$D_{P2P} \geq \max\left(\dfrac{F}{u_s}, \dfrac{F}{d_{min}}, \dfrac{NF}{u_s + \sum_{i=1}^{N} u_i}\right)$

[assume $d_{min} \gg u_s \gg u_i = u$]

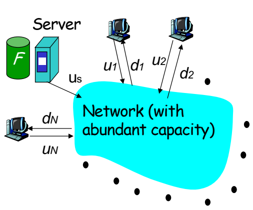
$\rightarrow \dfrac{\cancel{N}F}{u_s + \cancel{N}u}$

$\lim_{N \to \infty} D_{P2P} \rightarrow \dfrac{F}{u}$

71

File distribution time: client-server

- server sequentially sends N copies:
  - NF/u_s time
- client i takes F/d_i time to download



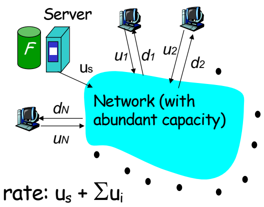
Time to distribute F to N clients using client/server approach $= d_{cs} \geq \max_i \{ NF/u_s, F/\min(d_i) \}$

increases linearly in N (for large N)

72

File distribution time: P2P

- server must send one copy: F/u_s time
- client i takes F/d_i time to download
- NF bits must be downloaded (aggregate)
  - fastest possible upload rate: $u_s + \Sigma u_i$



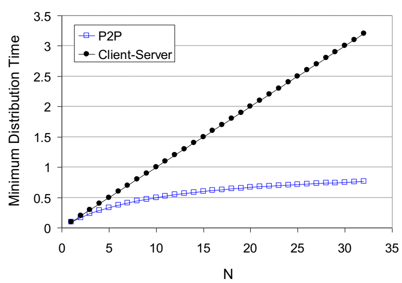
$$d_{P2P} >= \max\{ F/u_s, F/\min_i(d_i), NF/(u_s + \Sigma u_i) \}$$

Matta @ BUCS - Applications 1-73

73

Server-client vs. P2P: example

Client upload rate = u, F/u = 1 hour, u_s = 10u, $d_{min} \geq u_s$



Matta @ BUCS - Applications 1-74

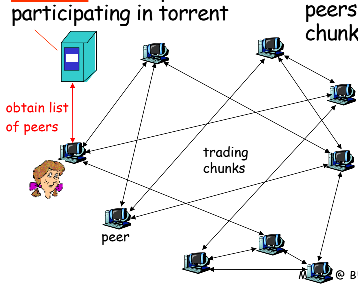
74

File distribution: BitTorrent

- P2P file distribution

tracker: tracks peers participating in torrent

torrent: group of peers exchanging chunks of a file



Seed: peer that has the whole file

Matta @ BUCS - Applications 1-75

75

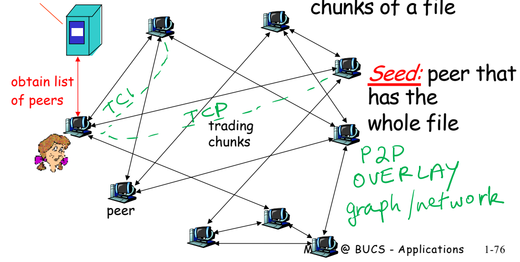
File distribution: BitTorrent

- P2P file distribution

torrent → IP address(es) of peers

tracker: tracks peers participating in torrent

torrent: group of peers exchanging chunks of a file



Seed: peer that has the whole file

P2P OVERLAY graph/network

76

BitTorrent (2)

Pulling Chunks

piece selection

- at any given time, different peers have different subsets of file chunks
- periodically, a peer (Alice) asks each neighbor for list of chunks that they have
- Alice sends requests for her missing chunks
  - rarest first

1,3,4
1,2,3
ASK (4)
1,2
missing 3&4

Sending Chunks: tit-for-tat

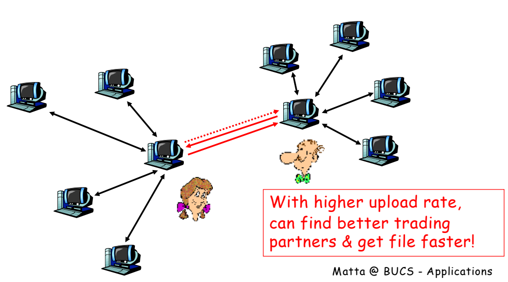
peer selection

- Alice sends chunks to four neighbors currently sending her chunks *at the highest rate*
  - re-evaluate top 4 every 10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - newly chosen peer may join top 4
  - "optimistically unchoke"

77

BitTorrent: Tit-for-tat

(1) Alice "optimistically unchokes" Bob
(2) Alice becomes one of Bob's top-four providers; Bob reciprocates
(3) Bob becomes one of Alice's top-four providers

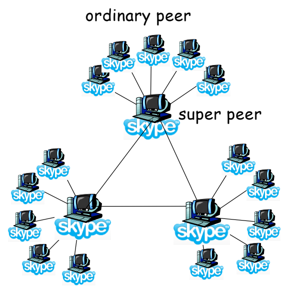


With higher upload rate, can find better trading partners & get file faster!

78

P2P Case study: Skype

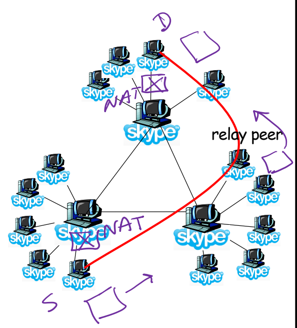
- proprietary application-layer protocol (inferred via reverse engineering)
- Hierarchical overlay of Skype peers
- Index maps usernames to IP addresses; distributed over super peers



79

Peers as relays

- Problem when both Alice and Bob are behind "NATs"
  - NAT prevents an outside peer from initiating a call to insider peer
- Solution:
  - Using Alice's and Bob's super peers, (non-NATed) Relay is chosen
  - Each peer initiates session with relay
  - Peers can now communicate through NATs via relay



80

Chapter 2: Summary

our study of network apps now complete!

- application architectures
  - client-server
  - P2P
  - hybrid
- application service requirements:
  - reliability, throughput, delay
- Internet transport service model
  - connection-oriented, reliable: TCP
  - unreliable, datagrams: UDP
- socket programming
- specific protocols:
  - HTTP
  - SMTP, POP, IMAP
  - DNS
  - P2P: BitTorrent, Skype, …

81

Chapter 2: Summary

Most importantly: learned about *protocols*

- typical request/reply message exchange:
  - client requests info or service
  - server responds with data, status code
- message formats:
  - headers: fields giving info about data
  - data: info being communicated

Important themes:

- persistent vs. non-persistent transport connections
- stateless vs. stateful
- caching
- reliable vs. unreliable msg transfer
- centralized vs. distributed
- Overlay vs. underlay