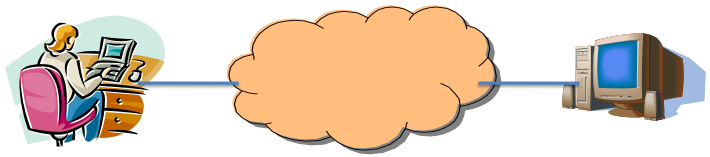
Content Distribution Networks + P2P File Sharing

COS 518: Advanced Computer Systems
Lecture 16

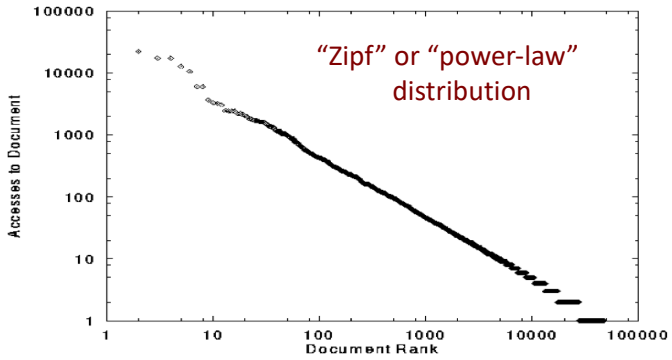
Mike Freedman

Single Server, Poor Performance



- Single server
  - Single point of failure
  - Easily overloaded
  - Far from most clients
- Popular content
  - Popular site
  - "Flash crowd"
  - Denial of Service attack

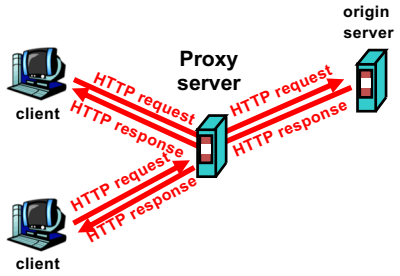
Skewed Popularity of Web Traffic



Characteristics of WWW Client-based Traces
Carlos R. Cunha, Azer Bestavros, Mark E. Crovella, BU-CS-95-01

Web Caching

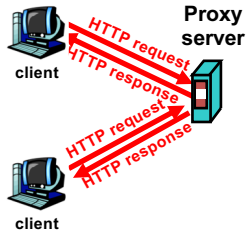
Proxy Caches



5

Forward Proxy

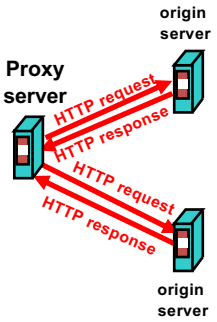
- Cache "close" to the client
  - Under administrative control of client-side AS
- Explicit proxy
  - Requires configuring browser
- Implicit proxy
  - Service provider deploys an "on path" proxy
  - … that intercepts and handles Web requests



6

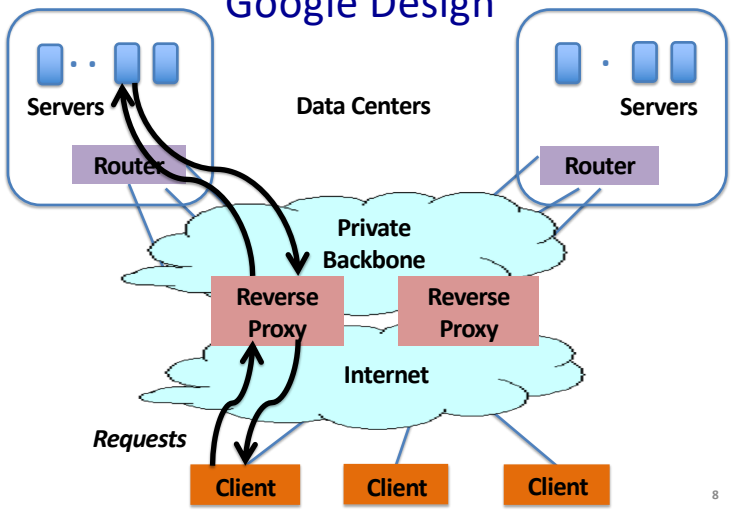
Reverse Proxy

- Cache "close" to server
  - Either by proxy run by server or in third-party CDNs
- Directing clients to the proxy
  - Map the site name to the IP address of the proxy



7

Google Design



8

Proxy Caches

(A) Forward (B) Reverse (C) Both (D) Neither

- Reactively replicates popular content
- Reduces origin server costs
- Reduces client ISP costs
- Intelligent load balancing between origin servers
- Offload form submissions (POSTs) and user auth
- Content reassembly or transcoding on behalf of origin
- Smaller round-trip times to clients
- Maintain persistent connections to avoid TCP setup delay (handshake, slow start)

Limitations of Web Caching

- Much content is not cacheable
  - Dynamic data: stock prices, scores, web cams
  - CGI scripts: results depend on parameters
  - Cookies: results may depend on passed data
  - SSL: encrypted data is not cacheable
  - Analytics: owner wants to measure hits
- Stale data
  - Or, overhead of refreshing the cached data

Modern HTTP Video-on-Demand

- Download "content manifest" from origin server
- List of video segments belonging to video
  - Each segment 1-2 seconds in length
  - Client can know time offset associated with each
  - Standard naming for different video resolutions and formats: e.g., 320dpi, 720dpi, 1040dpi, …
- Client downloads video segment (at certain resolution) using standard HTTP request.
  - HTTP request can be satisfied by cache: it's a static object
- Client observes download time vs. segment duration, increases/decreases resolution if appropriate

What about large files?

Peer-to-Peer Networks: BitTorrent

- BitTorrent history
  - 2002: B. Cohen debuted BitTorrent
- Emphasis on efficient fetching, not searching
  - Distribute same file to many peers
  - Single publisher, many downloaders
- Preventing free-loading
  - Incentives for peers to contribute

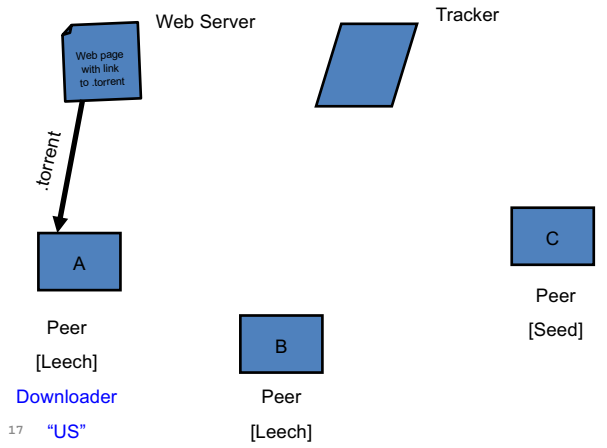
BitTorrent: Simultaneous Downloads

- Divide file into many chunks (e.g., 256 KB)
  - Replicate different chunks on different peers
  - Peers can trade chunks with other peers
  - Peer can (hopefully) assemble the entire file
- Allows simultaneous downloading
  - Retrieving different chunks from different peers
  - And uploading chunks to peers
  - Important for very large files

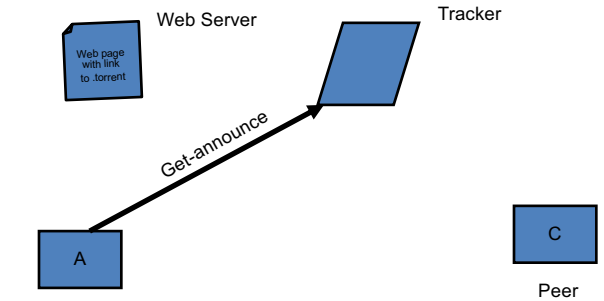
BitTorrent: Tracker

- Infrastructure node
  - Keeps track of peers participating in the torrent
  - Peers registers with the tracker when it arrives
- Tracker selects peers for downloading
  - Returns a random set of peer IP addresses
  - So the new peer knows who to contact for data
- Can have "trackerless" system
  - Using distributed hash tables (DHTs)

BitTorrent: Overall Architecture

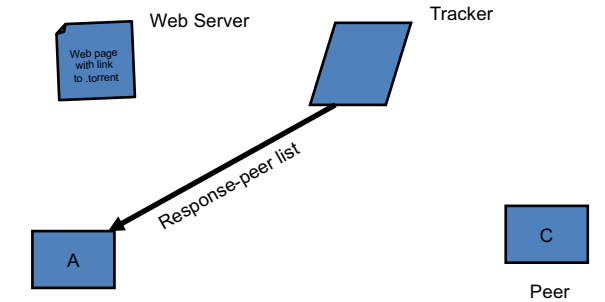


BitTorrent: Overall Architecture



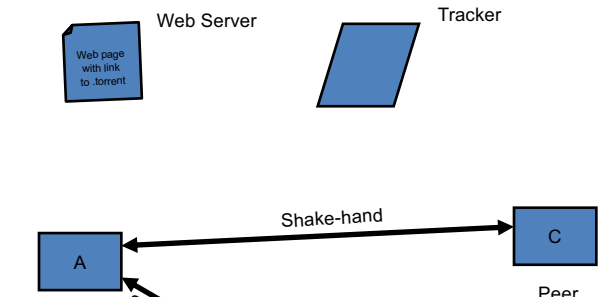
18

BitTorrent: Overall Architecture



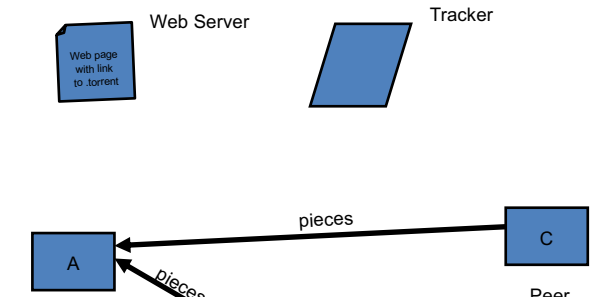
19

BitTorrent: Overall Architecture



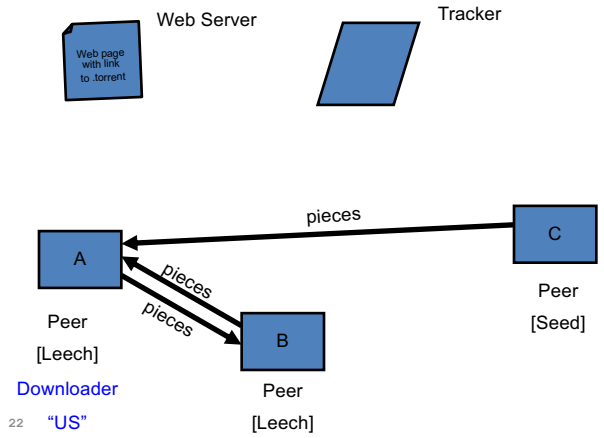
20

BitTorrent: Overall Architecture



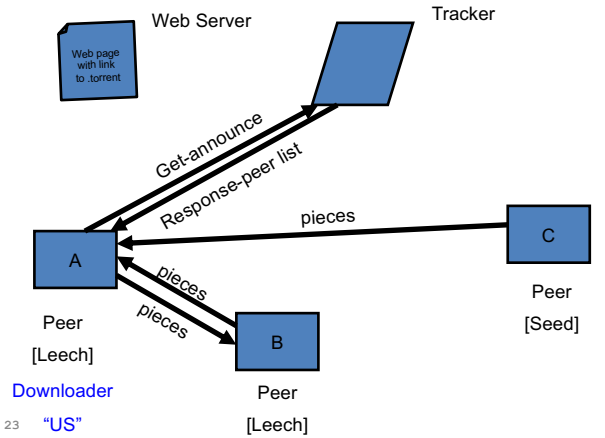
21

BitTorrent: Overall Architecture



22

BitTorrent: Overall Architecture



23

BitTorrent: Chunk Request Order

- Which chunks to request?
  - Could download in order
  - Like an HTTP client does
- Problem: many peers have the early chunks
  - Peers have little to share with each other
  - Limiting the scalability of the system
- Problem: eventually nobody has rare chunks
  - E.g., the chunks need the end of the file
  - Limiting the ability to complete a download
- Solutions: random selection and rarest first

24

BitTorrent: Rarest Chunk First

- Which chunks to request first?
  - Chunk with fewest available copies (i.e., rarest chunk)
- Benefits to the peer
  - Avoid starvation when some peers depart
- Benefits to the system
  - Avoid starvation across all peers wanting a file
  - Balance load by equalizing # of copies of chunks

25

Free-Riding in P2P Networks

- Vast majority of users are free-riders
  - Most share no files and answer no queries
  - Others limit # of connections or upload speed
- A few "peers" essentially act as servers
  - A few individuals contributing to the public good
  - Making them hubs that basically act as a server
- BitTorrent prevent free riding
  - Allow the fastest peers to download from you
  - Occasionally let some free loaders download

26

Bit-Torrent: Preventing Free-Riding

- Peer has limited upload bandwidth
  - And must share it among multiple peers
  - Tit-for-tat: favor neighbors uploading at highest rate
- Rewarding the top four neighbors
  - Measure download bit rates from each neighbor
  - Reciprocate by sending to the top four peers
- Optimistic unchoking
  - Randomly try a new neighbor every 30 seconds
  - So new neighbor has a chance to be a better partner

27

Conclusion

- Content distribution is hard
  - Many, diverse, changing objects
  - Clients distributed all over the world
  - Reducing latency is king
- Contribution distribution solutions
  - Reactive caching, proactive CDNs
- BitTorrent
  - Distributed download of large files
  - Anti-free-riding techniques
- Great example of how change can happen quickly in application-level protocols

28