CompSci 514: Computer Networks
Lecture 22: Content Distribution
Problem Statement

• One-to-many content distribution
Evolving Solutions

• Observation: duplicate copies of data are sent

• Solutions
  – IP multicast
  – End system multicast
  – Content distribution networks e.g. Akamai
  – P2P cooperative content distribution
    • Bittorrent etc.
IP multicast

- End systems join a multicast group
- Routers set up a multicast tree
- Packets are duplicated and forwarded to multiple next hops at routers
- Multicast pros and cons
Comments on IP Multicast

• Efficient
• Failed to deploy at the Internet scale
  – Added much state to routers → does not scale
  – Billing → Economic issues
  – Security
End system multicast

- End systems rather than routers organize into a tree, forward and duplicate packets
- Pros and cons
Comments on End System Multicast

• + Less infrastructure requirement
• - Single point of failure at the root node
• - Node joins and leaves causing much chain
• - One node may still be congested
• - Packets may traverse the same link twice
Content distribution networks

- Akamai
  - Works well but expensive, requires infrastructure support
Architecture components of a CDN

- Origin
- Overlay Routing
- Edge Server
- Client
- Authoritative Name Server (Global and Local Load Balancing)
- Content
- DNS
What is overlay routing?

- A direct end-to-end path is often sub-optimal in terms of latency, packet loss, and throughput.
- Solution → route packets through intermediate servers
Why is it called ”overlay”?

• Built on top of an underlay network
Dynamic File download speedup

Figure 10: A routing overlay provides significant speedups by choosing better performing paths from the origin to the client. Key: North America (NA), Europe (EU), Asia.
Catastrophe Insurance

Figure 11: Performance of the routing overlay during a cable cut.
Hierarchy for Scalability: tiered distribution

Entry point

Reflectors

...
Mapping an URL to an edge server

- Using DNS canonical names
- Using IP anycast
Canonical names and aliases

- Hosts can have several names.
- One is called canonical names and others are called aliases.

```plaintext
;; ANSWER SECTION:
prophet.cs.duke.edu. 86400 IN A 152.3.140.5
```
DNS redirection

• Using a hierarchy of DNS servers that translate a client’s web request to a nearby Akamai server
  – A client requests a DNS resolution (www.yahoo.com)
  – Akamai’s customer’s DNS name server uses a canonical name entry redirecting it to a DNS server in akamai’s network
  – A hierarchy of DNS name servers responds to the DNS name-translation request
  – Name of the Akamai customer and the name of the requested content as a guide to determine the best two Akamai edge servers
**CDNs Basics**

- Web client’s request redirected to ‘close’ by server
  - Client gets web site’s DNS CNAME entry with domain name in CDN network
  - Hierarchy of CDN’s DNS servers direct client to 2 nearby servers
```
;; DiG 9.4.2-P2 <<>> images.pcworld.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29098
;; flags: qr rd ra
QUERY: 1, ANSWER: 4, AUTHORITY: 9, ADDITIONAL: 2

;; QUESTION SECTION:
;images.pcworld.com. IN A

;; ANSWER SECTION:
images.pcworld.com.edgesuite.net. 21585 IN CNAME a1694.g.akamai.net.
a1694.g.akamai.net. 5 IN A 128.109.34.38
a1694.g.akamai.net. 5 IN A 128.109.34.45

;; AUTHORITY SECTION:
g.akamai.net. 973 IN NS n1g.akamai.net.
g.akamai.net. 973 IN NS n2g.akamai.net.
g.akamai.net. 973 IN NS n3g.akamai.net.
g.akamai.net. 973 IN NS n4g.akamai.net.
g.akamai.net. 973 IN NS n5g.akamai.net.
g.akamai.net. 973 IN NS n6g.akamai.net.
g.akamai.net. 973 IN NS n7g.akamai.net.
g.akamai.net. 973 IN NS n8g.akamai.net.
g.akamai.net. 973 IN NS n0g.akamai.net.

;; ADDITIONAL SECTION:
n1g.akamai.net. 1663 IN A 97.65.135.156
n5g.akamai.net. 889 IN A 128.109.247.10

;; Query time: 1 msec
;; SERVER: 152.3.140.1#53(152.3.140.1)
;; WHEN: Mon Feb 23 18:05:12 2009
;; MSG SIZE  rcvd: 337
```
Peer-to-Peer Cooperative Content Distribution

- Use the client’s uplink bandwidth

- New problem: incentives for cooperation or how to motivate clients to upload
The Gnutella approach

- **All nodes are true peers**
  - A peer is the publisher, the uploader and the downloader.
- **No single point of failure.**

- **Efficiency and scalability issue:**
  - File searches span across a large number of nodes generating lots of traffic.

- **Integrity, i.e. content pollution issue:**
  - Anyone can claim that he publishes valid content
  - No guarantee of quality of objects

- **Incentive issue:**
  - No incentives for cooperation (free-riding in Gnutella)
Peer-to-peer content distribution

- Centralized approaches
- The BitTorrent approach
BitTorrent overview

- File is divided into **chunks** (e.g. 256KB)
  - ShA1 hashes of all the pieces are included in the .torrent file for integrity check
  - A chunk is divided into sub-pieces to improve efficiency
- **Seeders** have all chunks of the file
- **Leechers** have some or no chunks of the file
BitTorrent overview

File is divided into **chunks** (e.g. 256KB)

- ShA1 hashes of all the pieces are included in the .torrent file for integrity check

- .torrent file has address of a tracker

- Tracker tracks all downloading clients
Terminology

- **Seeder**: peer with the entire file
  - Original Seed: The first seed
- **Leecher**: peer that’s downloading the file
  - Fairer term might have been “downloader”
- **Sub-piece**: Further subdivision of a piece
  - The “unit for requests” is a subpiece
  - But a peer uploads only after assembling complete piece
BitTorrent overview

- Clients (seeders or leechers) contact the **tracker**
- Tracker has complete **view** of the swarm
BitTorrent overview

- Tracker sends **partial view** to clients
- Clients connect to peers in their partial view
BitTorrent overview

Every 10 sec, seeders sample their peers’ download rates

Seeders unchoke 4-10 interested fastest downloaders
BitTorrent overview

- A node announces available chunks to their peers
- **Leechers** request chunks from their peers (locally rarest-first)
BitTorrent overview

Rate-based tit-for-tat

- Every 30 sec, leechers optimistically unchoke 1-2 peers
- Optimistic unchoking helps in:
  - discover other faster peers and prompt them to reciprocate
  - Bootstrap new peers with no data to upload
BitTorrent overview

**Rate-based tit-for-tat**
- Every 30 sec, leechers optimistically unchoke 1-2 peers
- Every 10 sec, leechers sample their peers’ upload rates
BitTorrent overview

◆ Rate-based tit-for-tat
  - Every 30 sec, client **optimistically unchokes 1-2** peers
  - Every 10 sec, seeder samples its peers’ upload rates
  - **Leecher unchokes 4-10** fastest interested uploaders
  - Leecher chokes other peers
  - Q: Why does this algo encourage cooperation?
Scheduling: Choosing pieces to request

• **Rarest-first**: Look at all pieces at all peers, and request piece that’s owned by fewest peers
  – Increases diversity in the pieces downloaded
    • avoids case where a node and each of its peers have exactly the same pieces; increases throughput
  – Increases likelihood all pieces still available even if original seed leaves before any one node has downloaded entire file
Start time scheduling

• Random First Piece:
  – When peer starts to download, request random piece.
    • So as to assemble first complete piece quickly
    • Then participate in uploads
  – When first complete piece assembled, switch to rarest-first
Choosing pieces to request

- **End-game mode:**
  - When requests sent for all sub-pieces, (re)send requests to all peers.
  - To speed up completion of download
  - Cancel request for downloaded sub-pieces
Outline

- Cooperative Content Distribution
- The BitTorrent approach
- BitTyrant
BitTyrant: a strategic BT client

• Question: can a strategic peer game BT to significantly improve its download performance for the same level of upload contribution?

• Conclusion: incentives do not build robustness. Strategic peers can gain significantly in performance.
Key ideas

• Observations:
  – Choked as long as one is among the fastest peer set
  – High capacity peers equally split its upload rates to active set peers
  – Altruism comes from unnecessary contributions

• Strategies:
  – Maximize per connection download bandwidth
  – Maximize the number of reciprocating peers
  – Do not upload more than needed for reciprocation
BitTyrant strategy

- Rank peers by \( d_p/u_p \)
- Unchoke in decreasing order of peer ranking until upload capacity is saturated
- Assumption: data is always available, download is not limited by data scarcity
Challenges

• Determining $u_p$
  – Initialized with the expected equal split capacity obtained from measurement
  – Periodically update it
    • Increase multiplicatively if peer does not reciprocate
    • Decrease if peer does

• Estimating $d_p$
  – Measure from download rate
  – Estimate from peer announced block available rate $U: U/\text{ActiveSize}$

• Sizing the neighborhood
  – Request as fast as possible
For each peer $p$, maintain estimates of expected download performance $d_p$ and upload required for reciprocation $u_p$.

Initialize $u_p$ and $d_p$ assuming the bandwidth distribution in Figure 2.

$d_p$ is initially the expected equal split capacity of $p$.

$u_p$ is initially the rate just above the step in the reciprocation probability.

Each round, rank order peers by the ratio $d_p/u_p$ and unchoke those of top rank until the upload capacity is reached.

\[
\frac{d_0}{u_0}, \frac{d_1}{u_1}, \frac{d_2}{u_2}, \frac{d_3}{u_3}, \frac{d_4}{u_4}, \ldots
\]

choose $k$ with $\sum_{i=0}^{k} u_i \leq cap$

At the end of each round for each unchoked peer:

If peer $p$ does not unchoke us: $u_p \leftarrow (1 + \delta)u_p$

If peer $p$ unchokes us: $d_p \leftarrow$ observed rate.

If peer $p$ has unchoked us for the last $r$ rounds: $u_p \leftarrow (1 - \gamma)u_p$

Figure 9: BitTyrant unchoke algorithm
BitTyrant Modelling

• **Reciprocation probability:**
  – Probability that a peer will select the client with certain equal split as downloader in the next unchoking round

• Active set: peers that upload to us
• Neighborhood: peers that we connect to
## Notations

<table>
<thead>
<tr>
<th>Label</th>
<th>Definition</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\omega$</td>
<td>2</td>
<td>Number of simultaneous optimistic unchokes per peer</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>80</td>
<td>Local neighborhood size (directly connected peers)</td>
</tr>
<tr>
<td>$b(r)$</td>
<td>Figure 1</td>
<td>Probability of upload capacity rate $r$</td>
</tr>
<tr>
<td>$B(r)$</td>
<td>$\int_0^r b(r),dr$</td>
<td>Cumulative probability of a upload capacity rate $r$</td>
</tr>
<tr>
<td>active($r$)</td>
<td>$\lfloor \sqrt{0.6r} \rceil - \omega$</td>
<td>Size (in peers) of the active transfer set for upload capacity rate $r$</td>
</tr>
<tr>
<td>split($r$)</td>
<td>$\frac{r}{\text{active}(r)+\omega}$</td>
<td>Per-connection upload capacity for upload capacity rate $r$</td>
</tr>
<tr>
<td>$s(r)$</td>
<td>Figure 1</td>
<td>Probability of an equal split rate $r$ using mainline active($r$) sizing</td>
</tr>
<tr>
<td>$S(r)$</td>
<td>$\int_0^r s(r),dr$</td>
<td>Cumulative probability of an equal-split rate $r$</td>
</tr>
</tbody>
</table>
Expected Reciprocation Probability

Upload / download: Probability of reciprocation for a peer $P$ with upload capacity $r_P$ from $Q$ with $r_Q$:

$$p_{recip}(r_P, r_Q) = 1 - (1 - S(r_P))^{\text{active}(r_Q)} \quad (1)$$

Expected reciprocation probability for capacity $r$:

$$\text{recip}(r) = \int b(x)p_{recip}(r, x)dx \quad (2)$$
Expected Download Rate

Expected download and upload rate for capacity $r$:

$$D(r) = \text{active}(r) \left[ \int b(x)p_{\text{recip}}(r, x)\text{split}(x)dx \right] +$$

$$\omega \left[ \int b(x)\text{split}(x)dx \right] \quad (3)$$

$$U(r) = \min \left( r, (\text{active}(r) + \omega)D(r) \right) \quad (4)$$
Summary

• Bittorrent
  – Swarm-based distribution
  – Tit-for-tat for incentives

• Bittyrant: a selfish client
  – Don’t upload faster than necessary
  – Estimate minimum upload rates to warrant unchoking