BitTorrent

CS514

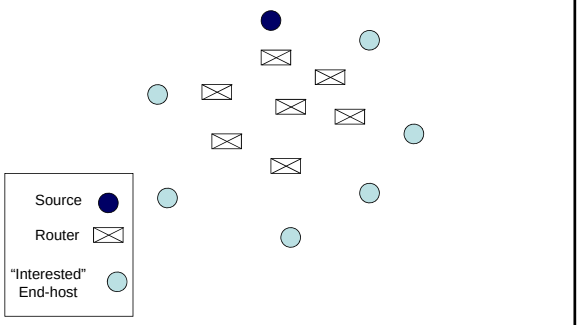
Vivek Vishnumurthy, TA

Common Scenario

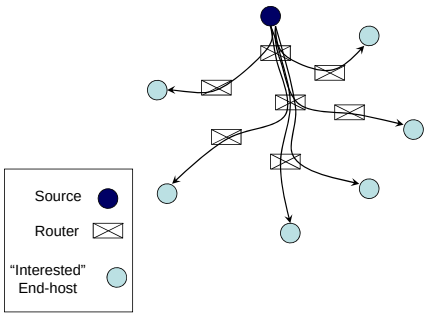
- Millions want to download the same popular huge files (for free)
  - ISO's
  - Media (the real example!)
- Client-server model fails
  - Single server fails
  - Can't afford to deploy enough servers

IP Multicast?

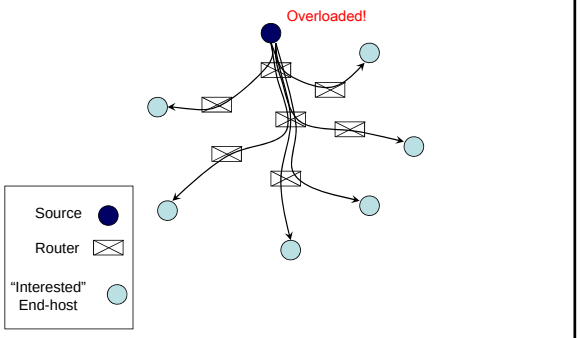
- Recall: IP Multicast not a real option in general settings
  - Not scalable
  - Only used in private settings
- Alternatives
  - End-host based Multicast
  - BitTorrent
  - Other P2P file-sharing schemes (later in lecture)



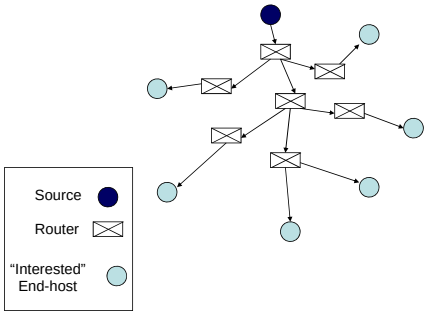
Client-Server



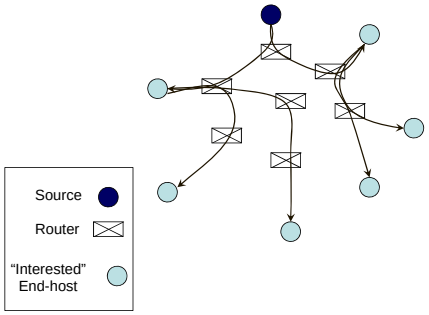
Client-Server



IP multicast



End-host based multicast



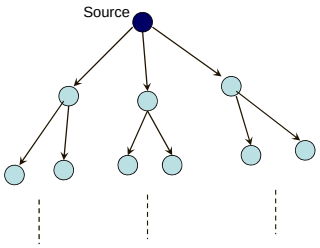
End-host based multicast

- "Single-uploader" ➔ "Multiple-uploaders"
  - Lots of nodes want to download
  - Make use of their *uploading* abilities as well
  - Node that has downloaded (part of) file will then upload it to other nodes.
  - ➢Uploading costs amortized across all nodes

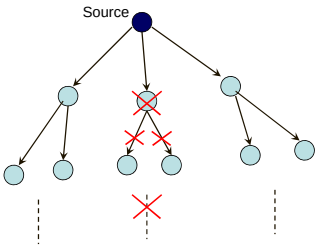
End-host based multicast

- Also called "Application-level Multicast"
- Many protocols proposed early this decade
  - Yoid (2000), Narada (2000), Overcast (2000), ALMI (2001)
    - All use single trees
    - Problem with single trees?

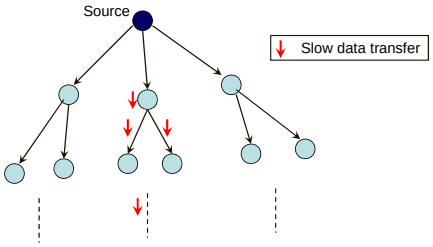
End-host multicast using single tree



End-host multicast using single tree



End-host multicast using single tree



End-host multicast using single tree

- Tree is “push-based” – node receives data, pushes data to children
- Failure of “interior”-node affects downloads in entire subtree rooted at node
- Slow interior node similarly affects entire subtree
- Also, leaf-nodes don’t do any sending!
- Though later multi-tree / multi-path protocols (Chunkyspread (2006), Chainsaw (2005), Bullet (2003)) mitigate some of these issues

BitTorrent

- Written by Bram Cohen (in Python) in 2001
- “Pull-based” “swarming” approach
  - Each file split into smaller pieces
  - Nodes request desired pieces from neighbors
    - As opposed to parents pushing data that they receive
  - Pieces not downloaded in sequential order
  - Previous multicast schemes aimed to support “streaming”; BitTorrent does not
- Encourages contribution by all nodes

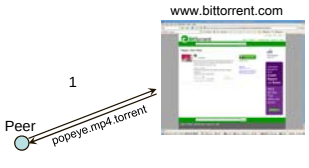
BitTorrent Swarm

- Swarm
  - Set of peers all downloading the same file
  - Organized as a random mesh
- Each node knows list of pieces downloaded by neighbors
- Node requests pieces it does not own from neighbors
  - Exact method explained later

How a node enters a swarm for file “popeye.mp4”

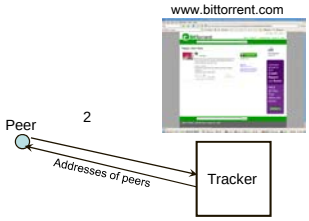
- File popeye.mp4.torrent hosted at a (well-known) webserver
- The .torrent has address of tracker for file
- The tracker, which runs on a webserver as well, keeps track of all peers downloading file

How a node enters a swarm for file “popeye.mp4”



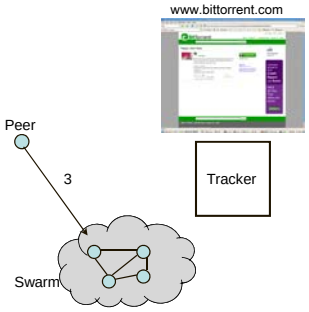
- File popeye.mp4.torrent hosted at a (well-known) webserver
- The .torrent has address of tracker for file
- The tracker, which runs on a webserver as well, keeps track of all peers downloading file

How a node enters a swarm for file "popeye.mp4"



- File popeye.mp4.torrent hosted at a (well-known) webserver
- The .torrent has address of tracker for file
- The tracker, which runs on a webserver as well, keeps track of all peers downloading file

How a node enters a swarm for file "popeye.mp4"



- File popeye.mp4.torrent hosted at a (well-known) webserver
- The .torrent has address of tracker for file
- The tracker, which runs on a webserver as well, keeps track of all peers downloading file

Contents of .torrent file

- URL of tracker
- Piece length – Usually 256 KB
- SHA-1 hashes of each piece in file
  - For reliability
- "files" – allows download of multiple files

Terminology

- Seed: peer with the entire file
  - Original Seed: The first seed
- Leech: 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

Peer-peer transactions: 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

Choosing pieces to request

- 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

Tit-for-tat as incentive to upload

- Want to encourage all peers to contribute
- Peer *A* said to choke peer *B* if it (*A*) decides not to upload to *B*
- Each peer (say *A*) unchokes at most 4 *interested* peers at any time
  - The three with the largest upload rates to *A*
    - Where the tit-for-tat comes in
  - Another randomly chosen (Optimistic Unchoke)
    - To periodically look for better choices

Anti-snubbing

- A peer is said to be snubbed if each of its peers chokes it
- To handle this, snubbed peer stops uploading to its peers

➢ Optimistic unchoking done more often

- Hope is that will discover a new peer that will upload to us

Why BitTorrent took off

- Better performance through "pull-based" transfer
  - Slow nodes don't bog down other nodes
- Allows uploading from hosts that have downloaded parts of a file
  - In common with other end-host based multicast schemes

Why BitTorrent took off

- Practical Reasons (perhaps more important!)
  - Working implementation (Bram Cohen) with simple well-defined interfaces for plugging in new content
  - Many recent competitors got sued / shut down
    - Napster, Kazaa
  - Doesn't do "search" per se. Users use well-known, trusted sources to locate content
    - Avoids the pollution problem, where garbage is passed off as authentic content

Pros and cons of BitTorrent

- Pros
  - Proficient in utilizing partially downloaded files
  - Discourages "freeloading"
    - By rewarding fastest uploaders
  - Encourages diversity through "rarest-first"
    - Extends lifetime of swarm
- Works well for "hot content"

Pros and cons of BitTorrent

- Cons
  - Assumes all interested peers active at same time; performance deteriorates if swarm "cools off"
  - Even worse: no trackers for obscure content

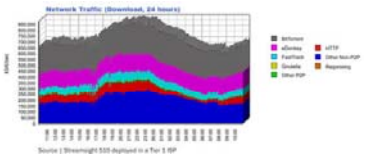
Pros and cons of BitTorrent

- Dependence on centralized tracker: pro/con?
  - ☹ Single point of failure: New nodes can't enter swarm if tracker goes down
  - Lack of a search feature
    - ☺ Prevents pollution attacks
    - ☹ Users need to resort to out-of-band search: well known torrent-hosting sites / plain old web-search

"Trackerless" BitTorrent

- To be more precise, "BitTorrent without a centralized-tracker"
- E.g.: Azureus
- Uses a Distributed Hash Table (Kademlia DHT)
- Tracker run by a normal end-host (not a web-server anymore)
  - The original seeder could itself be the tracker
  - Or have a node in the DHT randomly picked to act as the tracker

Why is (studying) BitTorrent important?



(From CacheLogic, 2004)

Why is (studying) BitTorrent important?

- BitTorrent consumes significant amount of internet traffic today
  - In 2004, BitTorrent accounted for 30% of all internet traffic (Total P2P was 60%), according to CacheLogic
  - Slightly lower share in 2005 (possibly because of legal action), but still significant
  - BT always used for legal software (linux iso) distribution too
  - Recently: legal media downloads (Fox)

Other file-sharing systems

- Prominent earlier: Napster, Kazaa, Gnutella
- Current popular file-sharing client: eMule
  - Connects to the ed2k and Kad networks
  - ed2k has a supernode-ish architecture (distinction between servers and normal clients)
  - Kad based on the Kademlia DHT

File-sharing systems…

- (Anecdotally) Better than BitTorrent in finding obscure items
- Vulnerable to:
  - Pollution attacks: Garbage data inserted with the same file name; hard to distinguish
  - Index-poisoning attacks (sneakier): Insert bogus entries pointing to non-existant files
  - Kazaa reportedly has more than 50% pollution + poisoning

References

- BitTorrent
  - "Incentives build robustness in BitTorrent", Bram Cohen
  - BitTorrent Protocol Specification: http://www.bittorrent.org/protocol.html
- Poisoning/Pollution in DHT's:
  - "Index Poisoning Attack in P2P file sharing systems"
  - "Pollution in P2P File Sharing Systems"