

18-345 – Fall 08

## Lecture 24: Peer-to-Peer

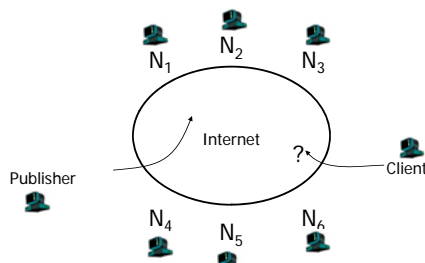
Peter Steenkiste

## Outline

- P2P file sharing techniques
  - Downloading: Whole-file vs. Chunks
  - Searching: Centralized index vs. Flooding
- Uses of P2P - what works well, what doesn't?
  - Servers vs. arbitrary nodes
  - Hard state (backups) vs. soft state (caches)
- Challenges
  - Fairness, freeloading, security, ...

2

## What is P2P?



The downloading process is *mainly* between peers

3

## Why P2P?

- Harness lots of spare capacity
  - 1 Big Fast Server: 1Gbit/s, \$10k/month++
  - 2,000 cable modems: 1Gbit/s, \$ ??
  - 1,000,000 end-hosts: Uh, wow
- Build self-managing systems that deal with huge scale
  - Same techniques attractive for both companies / servers / P2P
    - E.g., Akamai's 14,000 nodes
    - Google's 100,000+ nodes

4

## P2P file-sharing

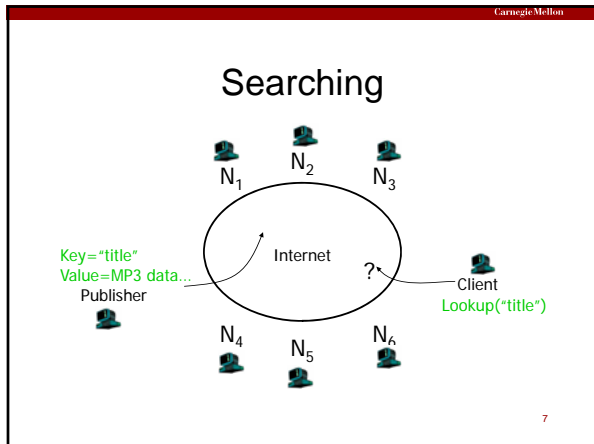
- Quickly grown in popularity
  - Dozens or hundreds of file sharing applications
  - 35 million American adults use P2P networks -- 29% of all Internet users in US!
  - Audio/Video transfer now dominates traffic on the Internet

5

## What's out there?

Search \ Download	Central	Flooding	Super-Node Flooding	Routing
Whole File	Napster	Gnutella		Freenet
Chunk Based	BitTorrent		KaZaA (bytes, not chunks)	DHTs eDonkey2000

6



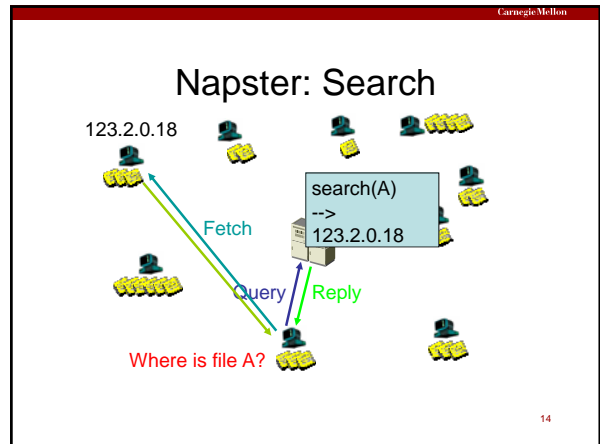
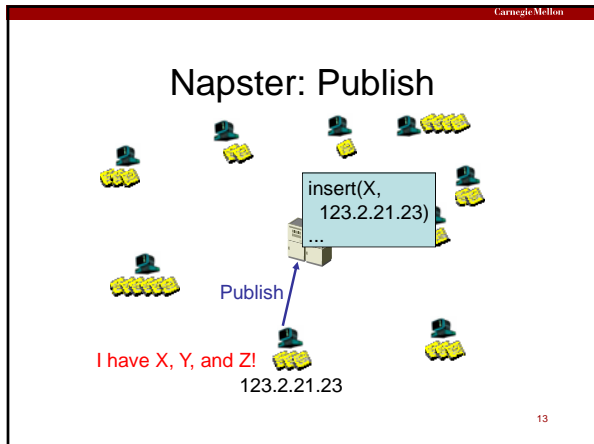
- Carnegie Mellon
- ## Searching (Cont.)
- Needles vs. Haystacks
    - Searching for popular files, or a file no one heard of
  - Search expressiveness
    - Whole word? Regular expressions? File names? Attributes? Whole-text search?
- 8

- Carnegie Mellon
- ## Framework
- Common Primitives:
    - **Join**: how do I begin participating?
    - **Publish**: how do I advertise my file?
    - **Search**: how do I find a file?
    - **Fetch**: how do I retrieve a file?
- 9

- Carnegie Mellon
- ## Next Topic...
- **Centralized Database**
    - Napster
  - **Query Flooding**
    - Gnutella
  - **Intelligent Query Flooding**
    - KaZaA
  - **Swarming**
    - BitTorrent
- 10

- Carnegie Mellon
- ## Napster: History
- 1999: Sean Fanning launches Napster
  - Peaked at 1.5 million simultaneous users
  - Jul 2001: Napster shuts down
    - Single point of lawsuit
- 11

- Carnegie Mellon
- ## Napster: Overview
- Centralized Database:
    - **Join**: contact central server
    - **Publish**: report list of files to server
    - **Search**: query the server => return someone that stores the requested file
    - **Fetch**: get the file directly from peer (using HTTP)
- 12

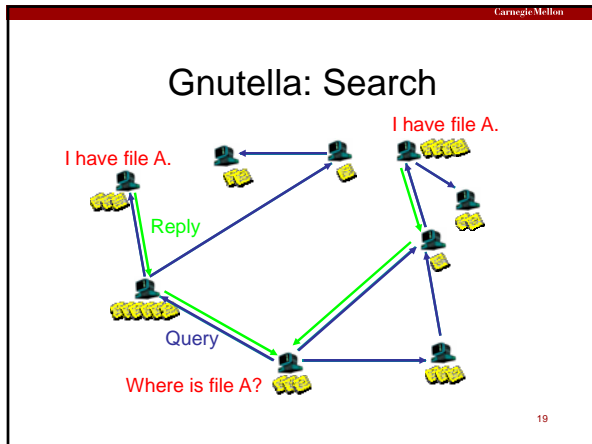


- Carnegie Mellon
- ## Napster: Discussion
- Pros:
    - Simple
    - Search scope is  $O(1)$
    - Controllable (pro or con?)
  - Cons:
    - Server maintains  $O(N)$  State
    - Server does all processing
    - Someone needs to administer it
    - Single point of failure
- 15

- Carnegie Mellon
- ## Next Topic...
- **Centralized Database**
    - Napster
  - **Query Flooding**
    - Gnutella
  - **Intelligent Query Flooding**
    - KaZaA
  - **Swarming**
    - BitTorrent
- 16

- Carnegie Mellon
- ## Gnutella: History
- In 2000, J. Frankel and T. Pepper from Nullsoft released Gnutella
  - Soon many other clients: Bearshare, Morpheus, LimeWire, etc.
  - In 2001, many protocol enhancements including "ultrapeers"
- 17

- Carnegie Mellon
- ## Gnutella: Overview
- Query Flooding:
    - **Join**: contact few nodes
      - These become "neighbors"
    - **Publish**: no need
    - **Search**: ask neighbors, who ask their neighbors, and so on... when/if found, reply to sender
      - TTL limits propagation
    - **Fetch**: get the file directly from peer
- 18

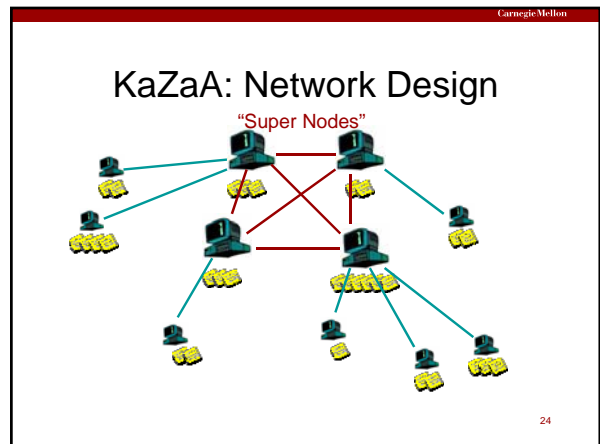


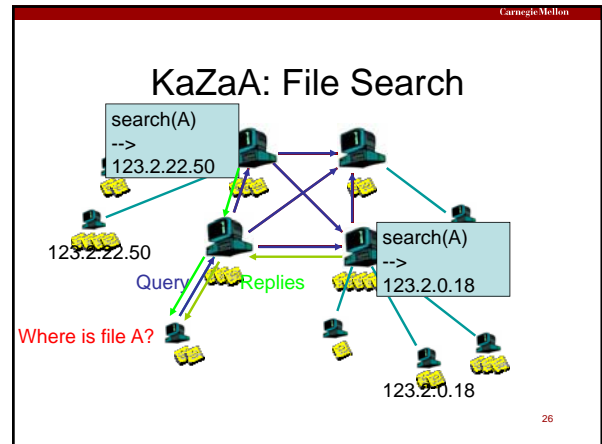
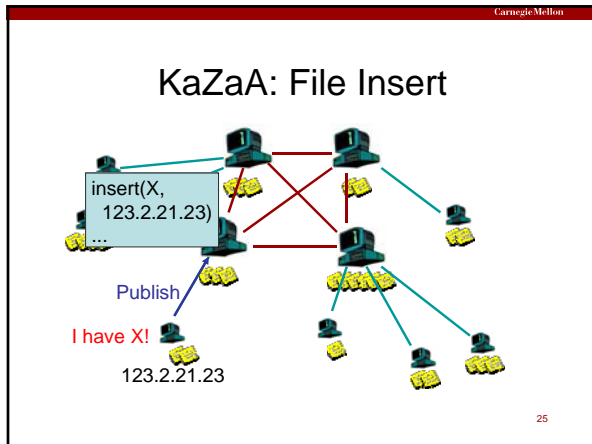
- Carnegie Mellon
- ## Gnutella: Discussion
- Pros:
    - Fully de-centralized
    - Search cost distributed
    - Processing @ each node permits powerful search semantics
  - Cons:
    - Search scope is  $O(N)$
    - Search time is  $O(???)$
    - Nodes leave often, network unstable
  - TTL-limited search works well for haystacks.
    - For scalability, does NOT search every node
    - May have to re-issue query later
- 20

- Carnegie Mellon
- ## Next Topic...
- **Centralized Database**
    - Napster
  - **Query Flooding**
    - Gnutella
  - **Intelligent Query Flooding**
    - KaZaA
  - **Swarming**
    - BitTorrent
- 21

- Carnegie Mellon
- ## KaZaA: History
- In 2001, KaZaA created by Dutch company KaZaA BV
  - Single network called FastTrack used by other clients as well: Morpheus, giFT, etc.
  - Eventually protocol changed so other clients could no longer talk to it
- 22

- Carnegie Mellon
- ## KaZaA: Overview
- “Smart” Query Flooding:
    - **Join**: contact a “super node” ... may at some point become one itself
    - **Publish**: send list of files to super node
    - **Search**: send query to super nodes, super nodes flood query amongst themselves
    - **Fetch**: get the file directly from peer(s); can fetch simultaneously from multiple peers
- 23





- Carnegie Mellon
- ## KaZaA: Fetching
- More than one node may have requested file...
  - How to tell?
    - Must be able to distinguish identical files
    - Not necessarily same filename
    - Same filename not necessarily same file...
  - Use Hash of file
    - KaZaA uses UUHash: fast, but not secure
    - Alternatives: MD5, SHA-1
  - How to fetch?
    - Get bytes [0..1000] from A, [1001...2000] from B
    - Alternative: Erasure Codes
- 27

- Carnegie Mellon
- ## KaZaA: Discussion
- Similar behavior to Gnutella, but better
  - Pros:
    - Tries to take into account node heterogeneity:
      - Bandwidth
      - Host computational resources
      - Host availability (?)
    - Rumored to take into account network locality
  - Cons:
    - Still not scalable when the number of super nodes increase
    - Still no real guarantees on search scope or search time
- 28

- Carnegie Mellon
- ## Stability and Super peers
- Why super peers?
    - Query consolidation
      - Many connected nodes may have only a few files
      - Propagating a query to a sub-node would take more b/w than answering it yourself
  - Super peer selection is time-based
    - How long you've been on is a good predictor of how long you'll be around
- 29

- Carnegie Mellon
- ## Next Topic...
- **Centralized Database**
    - Napster
  - **Query Flooding**
    - Gnutella
  - **Intelligent Query Flooding**
    - KaZaA
  - **Swarming**
    - BitTorrent
- 30

## BitTorrent: History

- In 2002, B. Cohen debuted BitTorrent
- Key Motivation:
  - Popularity exhibits temporal locality (Flash Crowds)
  - E.g., Slashdot effect, CNN on 9/11, new movie/game release
- Focused on Efficient *Fetching*, not *Searching*:
  - Distribute the *same* file to all peers
  - Gives an **incentive** to share a file
  - Single publisher, multiple downloaders
- Has some “real” publishers:
  - Blizzard Entertainment uses it to distribute their games

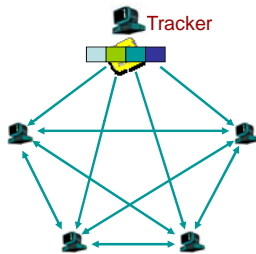
31

## BitTorrent: Overview

- **Swarming:**
  - **Join:** contact centralized “tracker” server
    - Get a list of peers
  - **Publish:** run a tracker server
  - **Search:** out-of-band
    - E.g., use Google to find a tracker for the file you want
  - **Fetch:** download chunks of the file from your peers and upload chunks you have to them
- **Big differences from Napster:**
  - Chunk based downloading
  - “few large files” focus
  - Anti-freeloading mechanisms

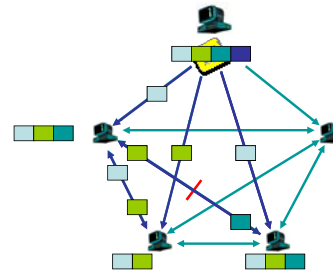
32

## BitTorrent: Publish/Join



33

## BitTorrent: Fetch



34

## BitTorrent: Sharing Strategy

- Employ “Tit-for-tat” sharing strategy
  - A is downloading from some other people
    - A will let the fastest N of those download from him
  - Be optimistic: occasionally let freeloaders download
    - Otherwise no one would ever start!
    - Also allows you to discover better peers to download from when they reciprocate
- Goal: Pareto Efficiency
  - Game Theory: “No change can make anyone better off without making others worse off”
  - Does it work? (don’t know!)

35

## BitTorrent: Summary

- **Pros:**
  - Works reasonably well in practice
  - Gives peers incentive to share resources; avoids freeloaders
- **Cons:**
  - Pareto Efficiency relative weak condition
  - Central tracker server needed to bootstrap swarm
    - Tracker is a design choice, not a requirement

36

## Writable, persistent P2P

- Do you trust your data to 100,000 monkeys?
- Node availability hurts
  - Ex: Store 5 copies of data on different nodes
  - When someone goes away, you must replicate the data they held
  - Hard drives are \*huge\*, but cable modem upload bandwidth is tiny - perhaps 10 Gbytes/day
  - Takes many days to upload contents of 200GB hard drive. Very expensive leave/replication situation!

37

## P2P: Summary

- Many styles; remember pros and cons of each
  - centralized, flooding, swarming
- Used for sharing large files
- Lessons learned:
  - Single points of failure are very bad
  - Flooding messages to everyone is bad
  - Underlying network topology is important
  - Not all nodes are equal
  - Need incentives to discourage freeloading
  - Privacy and security are important

38