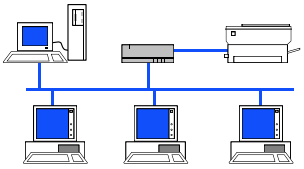


Distributed Systems 600.437 Peer to Peer Systems & Probabilistic Protocols

Department of Computer Science
The Johns Hopkins University

Peer to Peer Systems and Probabilistic Protocols

Lecture 11

Good reading:
Reliable Distributed Systems by Ken Birman - Chapter 25.

CACM article: http://cacm.acm.org/magazines/2010/10/99498-peer-to-peer-systems/fulltext

Peer to Peer

- What's in a name?
  - In contrast to "client server" systems.
  - A catchy name but not very meaningful. A lot of what we did in this course is actually peer communication between servers.
- A better name: client to client.
  - A different way to construct client-server systems where most, or all, of the server functionality resides on the clients themselves.

Peer to Peer (cont.)

- The promise:
  - Systems can be made much more scalable and reliable when the large number of clients are each contributing to the service.

Peer to Peer (cont.)

- The promise:
  - Systems can be made much more scalable and reliable when the large number of clients are each contributing to the service.
- Advantages:
  - Scalable to very large numbers (millions).
  - Stable under very high stress.
  - Self-repairing when disruptive failures occur.
- Issues to consider:
  - Churn: risk of melting down in case of rapid membership changes.
  - Tragedy of the common.

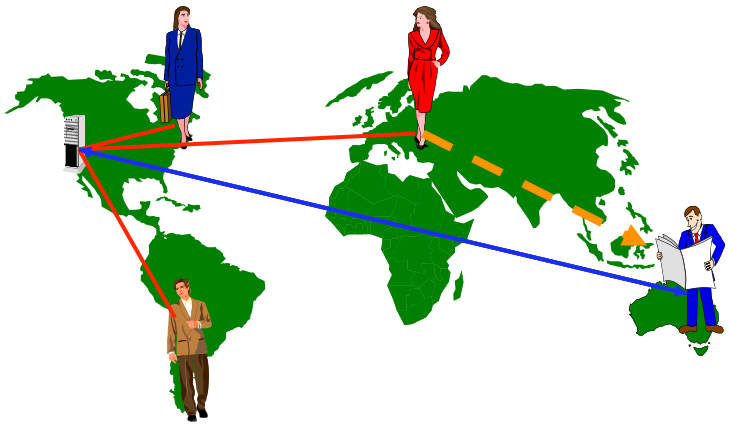
Peer to Peer File Sharing

- Peer to peer first application.
- A revolutionary way to distribute multimedia.
  - Extremely popular.
    - a million downloads of the Napster software per month in 2000.
    - Half a million simultaneous Napster users in 2000.
    - 100 Million BitTorrent users in 2011.
  - Also other kinds of files.
- Allows clients to:
  - **Share** their own files.
  - **Search** files in other clients' files.
  - **Download** other clients' files.

Napster

- Main idea – separation of lookup and service.
- Lookup is traditional.
  - Basically centralized lookup with some adaptations.
  - Clients register with centralized lookup service (Napster's site) and provide their available index.
  - Search is performed centrally.
  - Output of search includes the potential locations of requested file.
- File download is completely peer 2 peer.

Napster (cont.)



Napster Lessons

- It is amazing what can be done with one powerful centralized server (the lookup service) !!!
- Eventually, one centralized server could not keep up. They had to move to a centralized and regional servers structure.
- No control over the clients.
- Tragedy of the common:
  - Clients have incentive to utilize the system.
  - Clients have no incentive to contribute.
  - Clients that contribute have incentive to stop contributing.
  - As clients stop contributing the load on contributing clients goes up and their incentive to stop contributing goes up.
- It is ironic that Napster could be closed (legally) exactly because of the part that actually worked well – the centralized lookup.

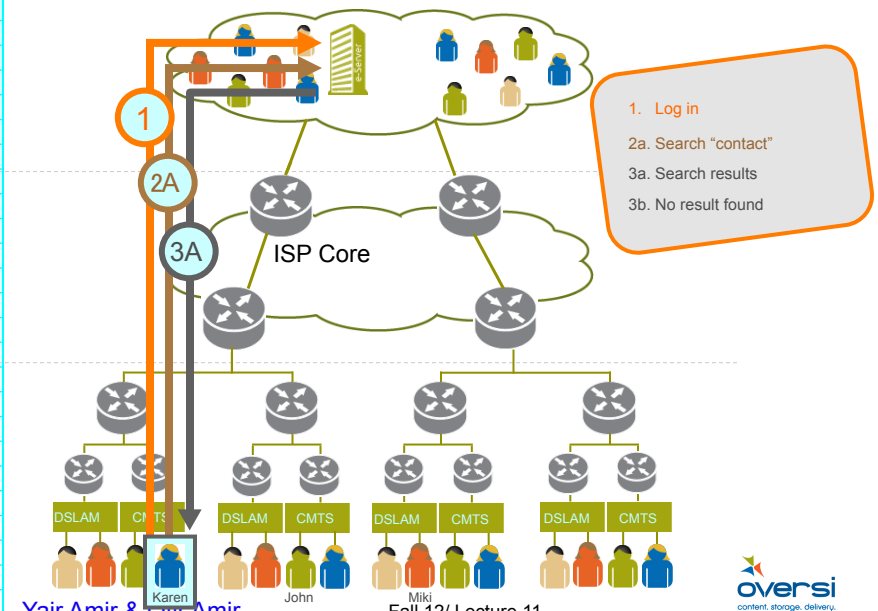
Genutella

- Eliminating the centralized lookup server:
  - To make it technically harder to close.
- Each client (a Gnutella node) is connected to a few other nodes.
- Each node updates its connection list as nodes come and go.
- Both lookup and service are distributed.
- Lookups are done in phases:
  - Each lookup phase is conducted as a controlled flood, limited by distance.
  - Flood distance is increased until desired file is located at least once.
- Pros / Cons ?

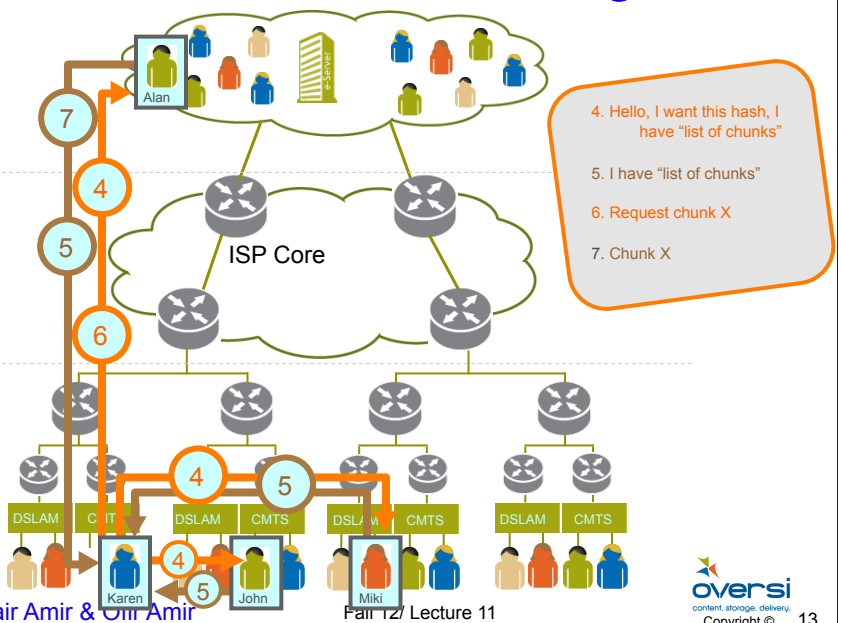
eD2K (eDonkey2000)

- First released in 2000 (eDonkey original client)
- The first major P2P network to support swarming
  - A file is logically divided into blocks.
  - Each block can be downloaded from a different source.
  - Multiple connections speed overall downloads for large files.
- MetaMachine, the company that developed the original eD2K network server and client software was shut down around 2005.
- The (current) servers are close-source freeware
- The eD2K network is still alive and kicking (mostly used in Europe)

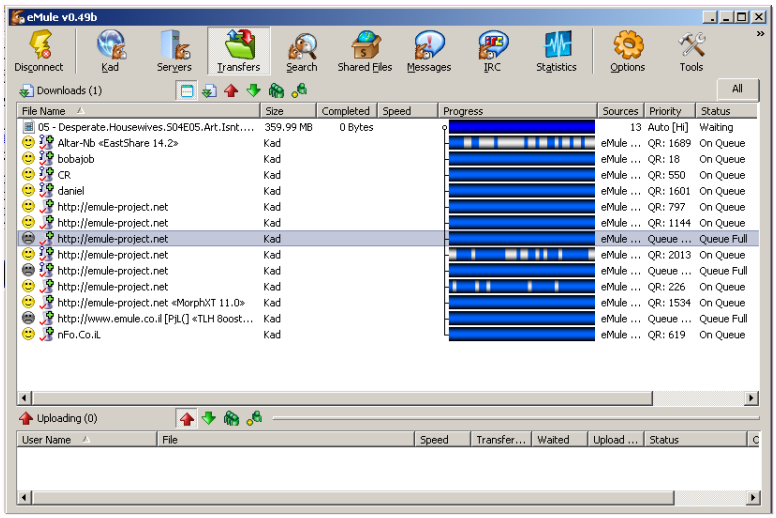
Ed2K Search and Connect



File Chunks Exchange



File Download in Action



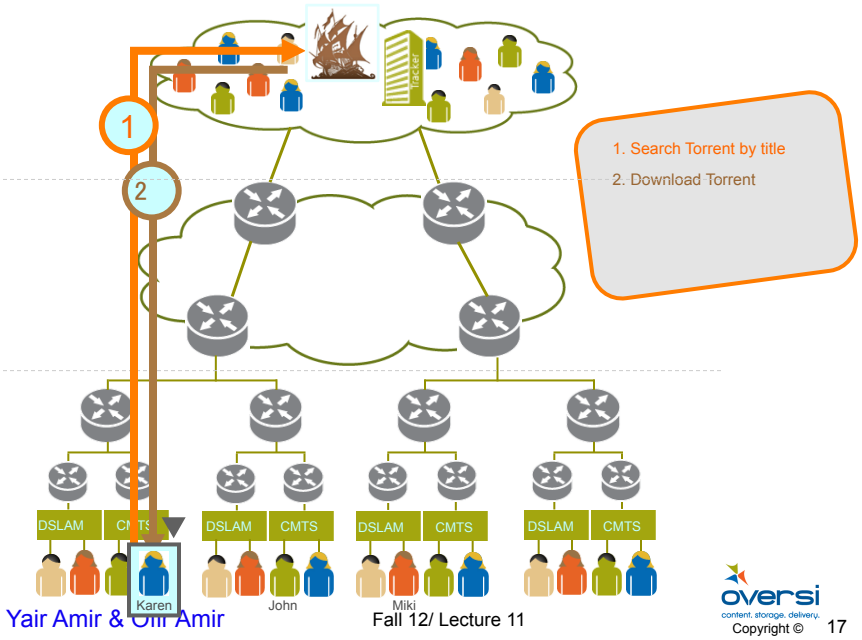
BitTorrent

- Protocol does not include content discovery.
  - Search for content is done outside of the protocol by other web-based means.
- Each file is an entity by itself.
  - Each file has a torrent file containing metadata about the file. Metadata includes:
    - Global hash for the file (SHA1).
    - Size, chunk size and hash for each chunk (SHA1).
    - List of trackers responsible for this file.
- Publisher of the file has to arrange for the trackers and the initial "seeder" – a server that has the file.
  - Computers that complete the download of the file serve as seeders.
  - Computers that are in the process of download are peers and can help others peers by providing blocks they already got

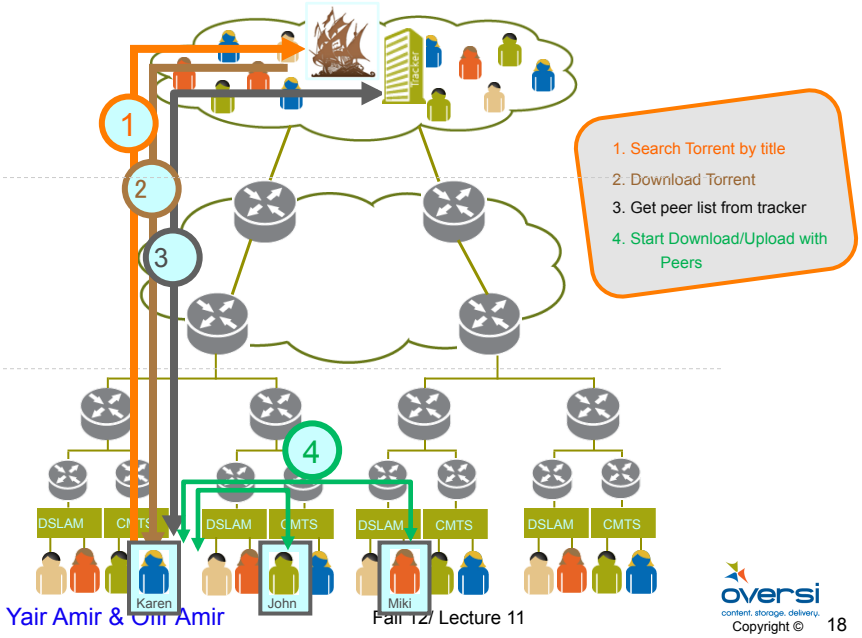
BitTorrent (cont)

- Swarm based download
  - Contact tracker to obtain list of peers (and seeders)
  - Contact peers and exchange information about which blocks each has
  - Request blocks you are missing from peers – each with its own TCP connection
  - While in the process, provide blocks you get requests for
- Block download
  - Request blocks according to Rarest First scheme
  - Verify each block based on hash in Torrent file
- Block uploads
  - Share blocks you have with other peers
  - When you have the complete file you become a seed

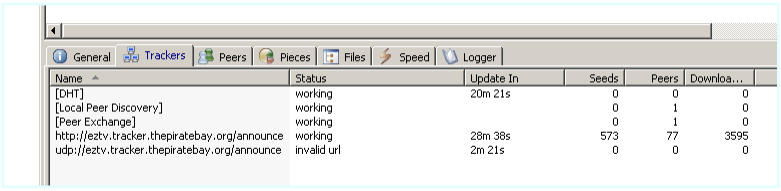
BitTorrent Search and Connect

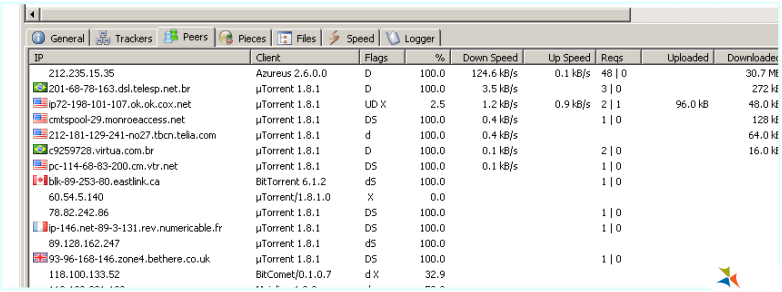


BitTorrent Search and Connect

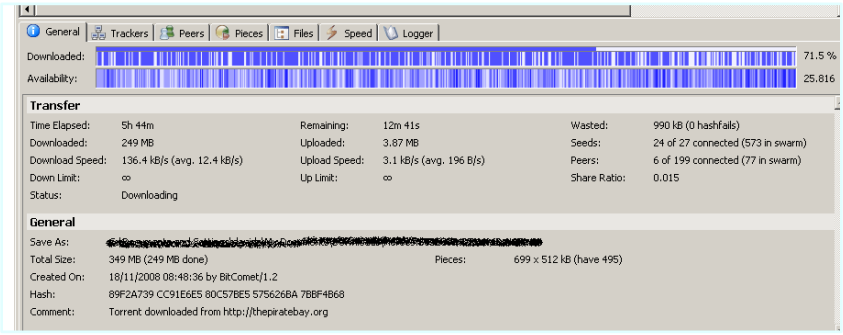


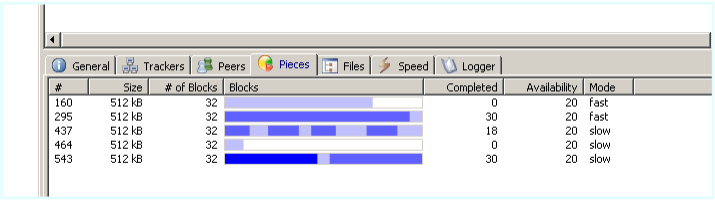
Trackers and Peer List (u torrent client screen)





File Download in Action





BitTorrent (cont)

- Other interesting points
  - Tit-for-tat sharing (with some capacity to go beyond that)
  - Super Seeding – give each peer a different block to better handle flash crowds and distribute the file (in addition to rarest-first requests)
  - Web Seeding – allow download from a web site using the HTTP protocol (to avoid the need for initial seeder)
  - Relatively low overhead (about 10%) compared with ED2K (with about 40%)
  - Incorporating Distributed Hash Table (DHT) methods to distribute the tracker and avoid this dependency in the protocol

Peer to Peer Distributed Indexing

- Distributed (and peer to peer) file systems have two parts:
  - A lookup mechanism that tracks down the node holding the object.
  - A superimposed file system application that actually retrieves and stores the files.
- Distributed indexing refers to the lookup part.
  - The **Internet DNS** is the most successful distributed indexing mechanism to date, mapping machine names to IP addresses.
  - Peer to peer indexing tries to generalize the concept to **(key, value)** pairs.
  - Also called Distributed Hash Table (**DHT**).

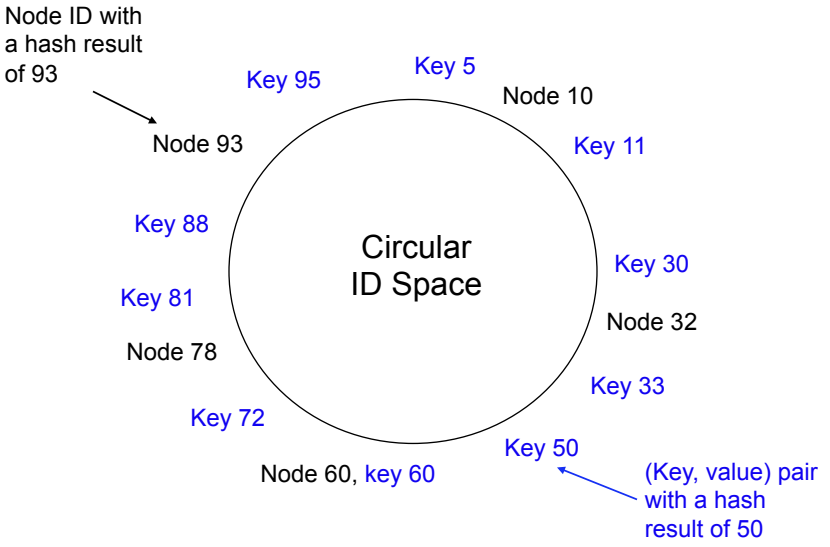
P2P Distributed Indexing (cont.)

- So, lets say we want to store a very large number of objects and access them based on their key.
- How would you implement a (key, value) distributed data structure that provides good performance for lookup and scales to a million nodes?
- …
- Now, think about what would you do to ensure robustness in the presence of participants coming and going.

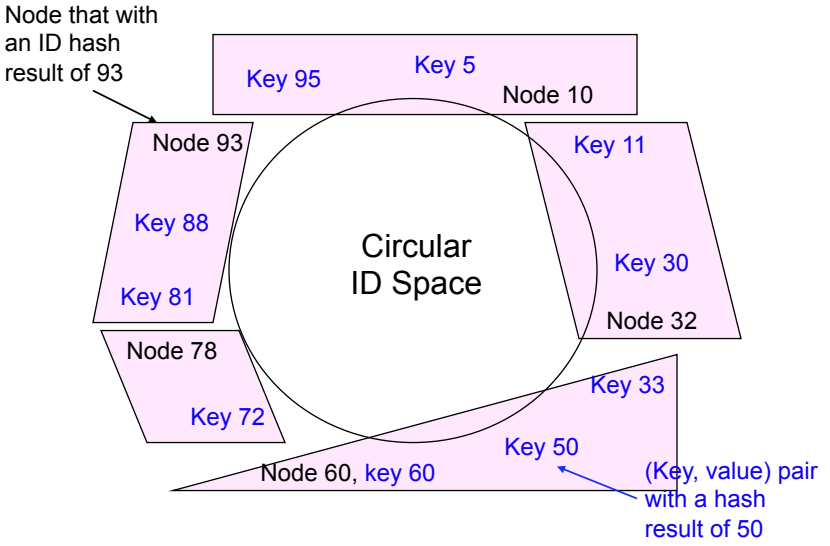
Chord

- Developed at MIT.
- Main idea: forming a massive virtual ring where every node is responsible for a portion of the periphery.
- Node IDs and data keys are hashed using the same function into a non-negative space.
- Each node is responsible for all the (**key,value**) pairs for which the hash result is less or equal to the node ID hash result, but greater then the next smaller hashed node ID.

Chord Indexing



Chord Indexing (cont.)



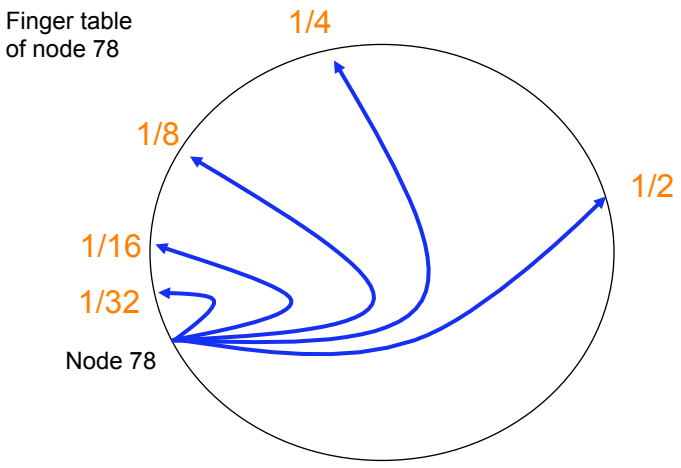
Chord Indexing (cont.)

- Each node maintains a pointer to the node after it and another pointer to the node before it.
- A new node contacts an existing node (startup issue) and traverses the ring until it finds the node before and after it.
- A state transfer is performed from the next node on the ring in order to accommodate the newly joined node.
- Lookup can be performed by traversing the ring, going one node at a time. Can we do better?

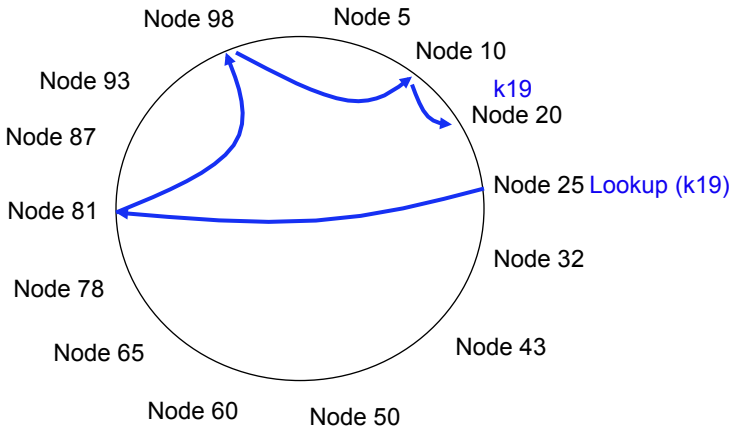
Chord Lookup

- Each node maintains a "finger table" that serves as short-cuts to nodes at various distances within the hash key space.
- Question:
  - How would you construct the "finger table" to allow logarithmic search steps?

Chord Lookup (cont.)



Chord Lookup (cont.)



Chord – Issues to Consider

- Overall, log(n) hops for lookup in the worst case! – very good.
- What is a hop? Where are the nodes? Is log(n) really good?
- What about churn?
- Is it really log(n) worst case over time?
- How to maintain robustness?

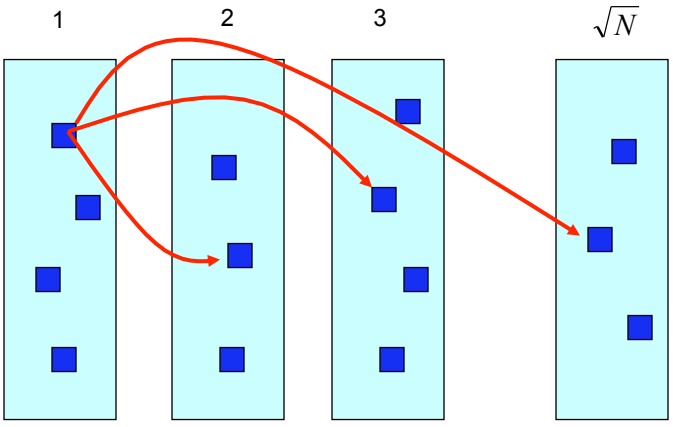
Kelips

- Developed at Cornell.
- Uses more storage (sqrt(n) instead of log(n)) at each node.
  - Replicating each item at sqrt(n) nodes.
- Aims to achieve O(1) for lookups.
- Copes with churn by imposing a constant communication overhead.
  - Although data quality may lag if updates occur too rapidly.
- How would you do that?

Kelips Lookup

- N is approximate number for the number of nodes.
- Each node id is hashed into one of sqrt(N) affinity groups.
- Each key from (**key,value**) pair is hashed into one of the sqrt(N) groups.
- Approximately sqrt(N) replicas in each affinity group.
- Pointers are maintained to a small number of members of each affinity group.
- Lookup is O(1).
- Weak consistency between the replicas is maintained using a reliable multicast protocol based on gossip.

Kelips Lookup (cont.)



Probabilistic Broadcast Protocols

- A class game demonstrating the probabilistic broadcast (pbcast) protocol:
  - At least n >= 20 logical participants.
  - Each participant randomly picks 3 numbers 1-n, noting the order of their selection.
  - Playing the game with the first number, then the first 2 numbers, then the 3 numbers and looking at coverage for a message generated by one participant.

P2P Impact

- Certainly very refreshing algorithms and interesting ways of thinking.
- Delivering on the promise.
  - Will P2P be able to scale beyond more traditional distributed servers approaches.
  - How will management and control issues be handled.
  - Will there be compelling applications.
- Or will it be another set of interesting techniques looking for a problem? My personal opinion:
  - Certainly has an impact (hence the need of ISPs to optimize)
  - What works on a large scale is managed services (think clouds). These will eat most of the P2P lunch for legal services.
  - Still some useful use in certain niches.