CPS 214:

Computer Networks and Distributed Systems

Networked Environments: Grid and P2P systems

Anda Iamnitchi
anda@cs.duke.edu

Class Objectives

- Start thinking of computer networking issue from the perspective of networked-applications
  - Because it’s more intuitive
  - Because it’s fun
- Understand some real applications in terms of:
  - Motivation, objectives
  - Resource/network requirements
  - Architecture (“distributed systems” part)

Grid and P2P Environments: Why

- Classes of applications rather than applications
- Principles rather than isolated solutions
- Popular (very, still) these days
- Deployed, real systems
- Eat up significant network resources
- Generate a lot of hype (and we must recognize it)
- Other applications are described in textbook(s)
  - Such as email, ftp, web, etc.

Outline

- Grid environments:
  - Characteristics
  - Case study: Grid2003
- Volunteer computing
  - Characteristics
  - Case study: Seti@home
- Peer-to-Peer
  - Characteristics (and some definitions)
  - Impact
  - Killer app: file sharing
  - Case study: Gnutella

Grids: Characteristics

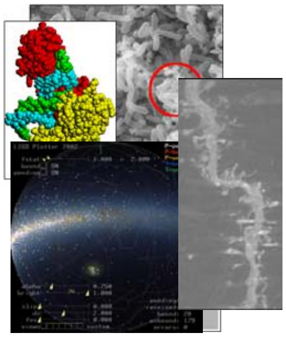
- Users:
  - Hundreds from 10s of institutions
  - Homogeneous, often trusted (well-established) communities
  - Implicit incentives for good behavior
- Resources:
  - Computers, data, instruments, storage, applications
  - Usually owned/administered by institutions
  - Highly available
- Typical applications: data- and compute-intensive processing
- Objective: common infrastructure for basic services
  - Resource discovery, Data management, Job management, Authentication, Monitoring, etc.

Case Study: Grid2003

- More than 25 U.S. LHC institutions, plus one Korean site.
- More than 2000 CPUs in total.
- More than 100 individuals authorized to use the Grid.
- Peak throughput of 500-900 jobs running concurrently, completion efficiency of 75%.

Grid2003 Applications

- 6 VOs, 11 Apps
- High-energy physics simulation and data analysis
- Cosmology based on analysis of astronomical survey data
- Molecular crystalography from analysis of X-ray diffraction data
- Genome analysis
- System “exercising” applications



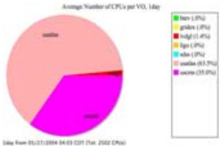
Grid2003 Applications

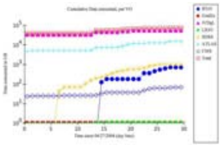
- CMS proton-proton collision simulation
- ATLAS proton-proton collision simulation
- LIGO gravitational wave search
- SDSS galaxy cluster detection
- ATLAS interactive analysis
- BTeV proton-antiproton collision simulation
- SnB biomolecular analysis
- GADU/Gnare: genome analysis
- Various computer science experiments

www.ivdgl.org/grid2003/applications

Grid2003 Interesting Points

- Each virtual organization includes its own set of system resources (compute nodes, storage, etc.) and people. VO membership info is managed system-wide, but policies are enforced at each site.
- *Throughput* is a key metric for success, and monitoring tools are used to measure it and generate reports for each VO.





Grid2003 Metrics

Metric	Target	Achieved
Number of CPUs	400	2762 (28 sites)
Number of users	> 10	102 (16)
Number of applications	> 4	10 (+CS)
Number of sites running concurrent apps	> 10	17
Peak number of concurrent jobs	1000	1100
Data transfer per day	> 2-3 TB	4.4 TB max

Outline

- Grid environments:
  - Characteristics
  - Case study: Grid2003
- Volunteer computing
  - Characteristics
  - Case study: Seti@home
- Peer-to-Peer
  - Characteristics (and some definitions)
  - Impact
  - Killer app: file sharing
  - Case study: Gnutella

Application: Number crunching

- Examples: Seti@Home, Entropia, UnitedDevices, DistributedScience, many others
- Incentives
  - ET cool, physics/AIDS research/etc. less so
- Approach suitable for a particular class of problems.
- Some characteristics (for Seti@Home):
  - Massive parallelism
  - Low bandwidth/computation ratio
  - Fixed-rate data processing task
  - Error tolerance
- Users do donate *real* resources
  - Why?

$1.5M / year
extra consumed power

SETI@home

- **SETI@home** "is a scientific experiment that uses Internet-connected computers in the Search for Extraterrestrial Intelligence (SETI). You can participate by running a free program that downloads and analyzes radio telescope data. "
- Is it Grid? Or P2P?
- Puts to work huge pool of underutilized resources

	Total	Last 24 Hours (as of 01/13/2005 am)
Users	5,315,717	1,193
Results received	1,724 millions	1.45 millions
Total CPU time	2.18 million years	1055.442 years
Average CPU time/work unit	11 hr 07 min 08.4 sec	6 hr 21 min 15.4 sec

Outline

- Grid environments:
  - Characteristics
  - Case study: Grid2003
- Volunteer computing
  - Characteristics
  - Case study: Seti@home
- Peer-to-Peer
  - Characteristics (and some definitions)
  - Impact
  - Killer app: file sharing
  - Case study: Gnutella

P2P Definition(s)

A number of definitions coexist:

- *Def 1:* "A class of applications that takes advantage of resources — storage, cycles, content, human presence — available at the edges of the Internet."
  - Edges often turned off, without permanent IP addresses
- *Def 2:* "A class of decentralized, self-organizing distributed systems, in which all or most communication is symmetric."
- Lots of other definitions that fit in between
- Lots of (P2P?) systems that fit nowhere…

P2P: Characteristics

- **Users:**
  - Millions
  - Anonymous individuals
  - No implicit incentives for good behavior (free riding, cheating)
- **Resources:**
  - Computing cycles XOR files
  - Resources owned/administered (?) by user
  - Intermittent (user/resource) participation:
    - Gnutella: average lifetime 60 min. ('01)
    - MojoNation: 1/6 users always connected ('01)
    - Overnet: 50% nodes available 70% of time over a week ('02)
- **Typical applications:** file retrieval or parallel computations
- **Vertically integrated solutions:**
  - Although signs of change: BOINC

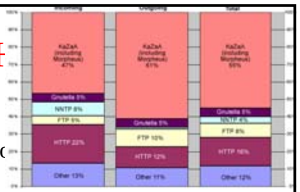
P2P Impact: Widespread adoption

- KaZaA – 170 millions downloads (3.5M/week) the most popular application ever!
- Number of users for file-sharing applications (www.slyck.com)

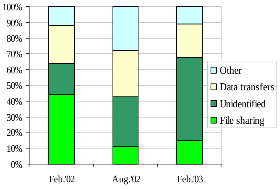
01/03/2005, 19:00)

eDonkey	2,698,388
FastTrack	2,065,657
Warez	1,402,729
Gnutella	1,115,086
OverNet	1,056,558
DirectConnect	273,485
MP2P	264,043

P2P Impact (2): F



- P2P generated traffic now dc load
  - Internet2 traffic statistics
  - Cornell.edu (March '02): 60% P2P
  - UChicago estimate (Marc control traffic about 1% o



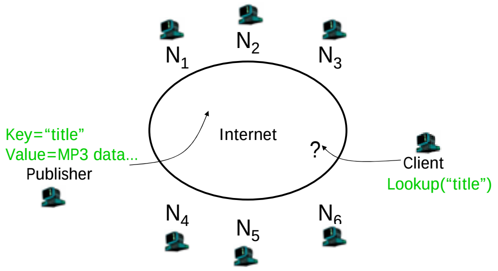
P2P Impact (3)

- Might force companies to change their business models
- Data copying and distribution carries almost zero cost now → this might impact copyright laws
- “New” research domain → grants and PhD theses

P2P killer app: File sharing

- Too many to list them all:
  - Napster, FastTrack (KaZaA, KazaaLite), Gnutella (LimeWire, Morpheus, BearShare), iMesh, eDonkey, MP2P, DirectConnect, Filetopia,
- New names/favorites appearing all the time
- Other app:
  - Instant messaging (Yahoo, AOL)
  - Collaborative environments (Groove)
  - Backup storage (HiveNet, OceanStore)
  - Spam filtering
  - Anonymous email
  - Censorship-resistant publishing systems (Ethernity, Freenet)
  - Content distribution
  - Network measurements

The Lookup Problem



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?

Outline

- Grid environments:
  - Characteristics
  - Case study: Grid2003
- Volunteer computing
  - Characteristics
  - Case study: Seti@home
- Peer-to-Peer
  - Characteristics (and some definitions)
  - Impact
  - Killer app: file sharing
  - Case study: Gnutella

P2P Case Study: Gnutella

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”

Gnutella Network

Why analyze Gnutella network?

- Large scale
  - up to 500k nodes, 100TB data, 10M files today
- Self-organizing network
- Fast growth in its early stages
  - more than 50 times during first half of 2001
- Open architecture, simple and flexible protocol
- Interesting mix of social and technical issues

Gnutella protocol overview

- P2P file sharing app. on top of an overlay network
  - Nodes maintain open TCP connections
  - Messages are broadcasted (flooded) or back-propagated
- (Initial) protocol

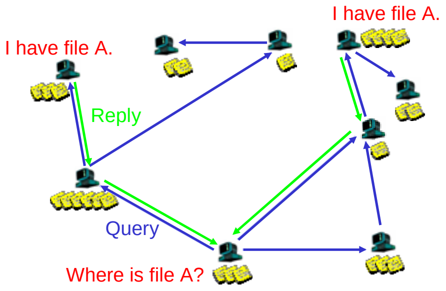
	Broadcast (Flooding)	Back-propagated	Node to node
Membership	PING	PONG	
Query	QUERY	QUERY HIT	
File download			GET, PUSH

- Protocol refinements (2001 and later)
  - Ping messages used more efficiently, Vendor specific extensions, GWebCaches, XML searches, super-nodes (2-layer hierarchy).

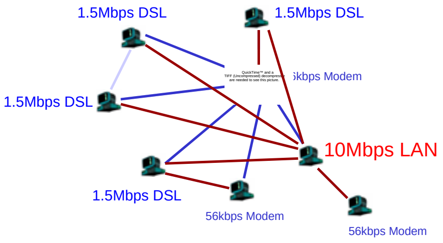
Gnutella: Overview

- Query Flooding:
  - **Join**: on startup, client contacts a few other nodes; these become its “neighbors”
  - **Publish**: no need
  - **Search**: ask neighbors, who is their neighbors, and so on... when/if found, reply to sender.
  - **Fetch**: get the file directly from peer

Gnutella: Search

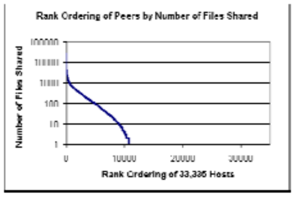


Gnutella: All Peers Equal? (1)



Gnutella: Free Riding
All Peers Equal? (2)

- More than 25% of Gnutella clients share no files; 75% share 100 files or less
- Conclusion: Gnutella has a high percentage of free riders
- If only a few individuals contribute to the public good, these few peers effectively act as centralized servers.



Adar and Huberman (Aug '00)

Gnutella: Discussion

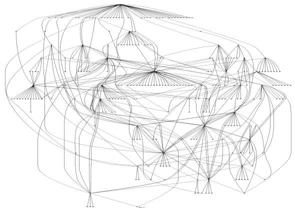
- Pros:
  - Fully de-centralized
  - Search cost distributed
- Cons:
  - Search scope is O(*N*)
  - Search time is O(???)
  - Nodes leave often, network unstable

What would you ask about Gnutella?

- Topology?
- Scale?
- Distribution of user requests?
- Node dynamics?
- Network traffic?
- …

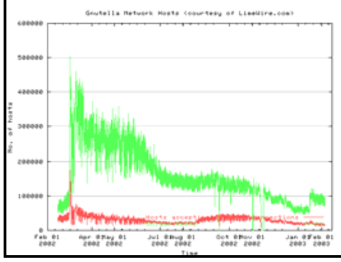
Gnutella:
Tools for Network Exploration

- ***Eavesdropper*** - modified node inserted into the network to log traffic.
- ***Crawler*** - connects to all active nodes and uses the membership protocol to discover graph topology.
  - *Client-server* approach.



Gnutella: Network Size

Explosive growth in 2001, but slowly shrinking thereafter

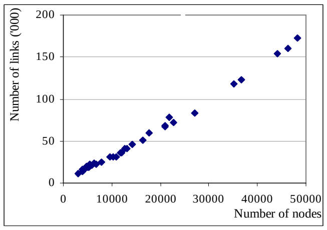


High user interest
- Users tolerate high latency, low quality results

Better resources
- DSL and cable modem nodes grew from 24% to 41% over first 6 months.

Gnutella: Growth Invariants

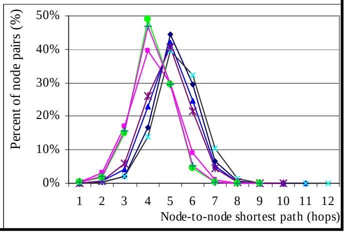
- (1) Unchanged average node connectivity
  - 3.4 links/node on average



Gnutella: Growth Invariants

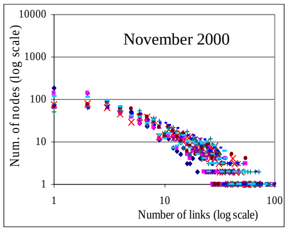
- (1) Unchanged average node connectivity
- (2) Node-to-node distance maintains similar distribution

Average node-to-node distance varied only 25% while the network grew 50 times over 6 months



Is Gnutella a power-law network?

Power-law networks: the number of links per node follows a power-law distribution $N = L^{-k}$



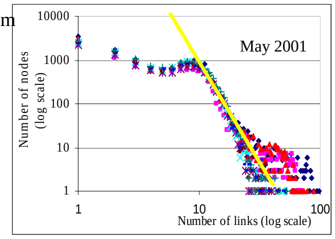
Examples:

- The Internet,
- In/out links to/from HTML pages,
- Citations network,
- US power grid,
- Social networks.

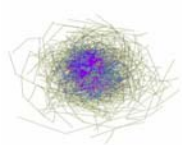
Implications: High tolerance to random node failure but low reliability when facing of an 'intelligent' adversary

Is Gnutella a power-law network?

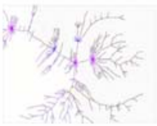
- Later, larger networks display a bimodal distribution
- *Implications*:
  - High tolerance to random node failures preserved
  - Increased reliability when facing an attack.



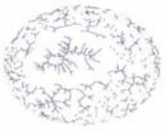
Network Resilience



Partial Topology



Random 30% die

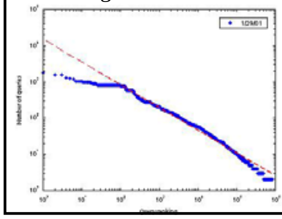


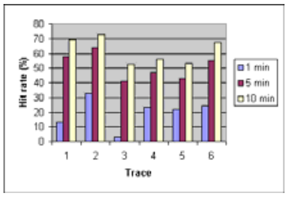
Targeted 4% die

from Saroiu *et al.*, *MMCN* 2002

Gnutella: Query distribution

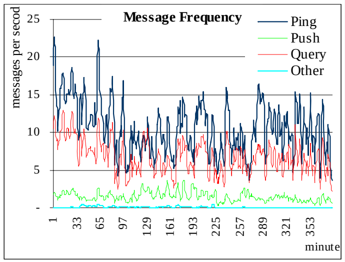
- Similar to Web pages popularity: Zipf distribution for query popularity
  - Significance: caching will work well
- Later results: in fact, not really Zipf (caching might not work that well)





Gnutella: Traffic analysis

- ≈ 6-8 kbps per link over all connections
- Traffic structure changed over time

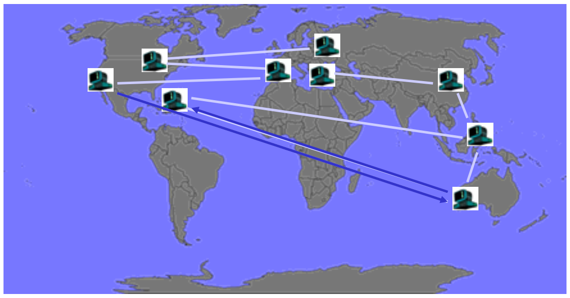


Gnutella:Total generated traffic

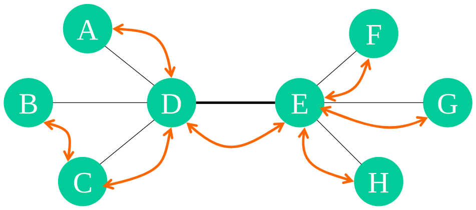
1Gbps (or 330TB/month)!

- Note that this estimate excludes actual file transfers
- Q: Does it matter?
- Compare to 15,000TB/month estimated in US Internet backbone (Dec. 2000)

Gnutella Topology Mismatch

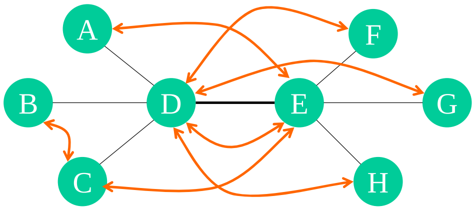


Gnutella: Topology Mismatch



Perfect mapping!

Gnutella: Topology Mismatch



- Inefficient mapping
- Link D-E needs to support six times higher traffic.

Gnutella: Topology Mismatch

The overlay network topology doesn’t match the underlying Internet infrastructure topology!

- 40% of all nodes are in the 10 largest Autonomous Systems (AS)
- Only 2-4% of all TCP connections link nodes within the same AS
- Largely ‘random wiring’

Gnutella: Summary

- Gnutella: self-organizing, large-scale, P2P application based on overlay network. It works!
- Discovered growth invariants specific to large-scale systems that:
  - Help predict resource usage.
  - Give hints for better search and resource organization techniques.
- Growth hindered by the volume of generated traffic and inefficient resource use.

P2P: Summary

- Many different styles; pros and cons of each
  - centralized, flooding, swarming, unstructured and structured routing
- 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
  - Structure can provide theoretical bounds and guarantees

Discussions

- Naturally emerging applications
  - Some more than others (science is more planned than music swapping, theoretically)
- Posing new challenges to the underlying network
  - Grids: QoS, payments OK, account for failures, fast reliable transfers, bulk data transfer
  - P2P: huge traffic, new policies for ISP (universities)
  - Volunteer computing: not much
- Examples of how applications shape the network protocols.

Questions?

DHT: History

- In 2000-2001, academic researchers said “we want to play too!”
- Motivation:
  - Frustrated by popularity of all these “half-baked” P2P apps :)
  - We can do better! (so we said)
  - Guaranteed lookup success for files in system
  - Provable bounds on search time
  - Provable scalability to millions of node
- Hot Topic in networking ever since

DHT: Overview

- **Abstraction**: a distributed “hash-table” (DHT) data structure:
  - put(id, item);
  - item = get(id);
- **Implementation**: nodes in system form a distributed data structure
  - Can be Ring, Tree, Hypercube, Skip List, Butterfly Network, ...

DHT: Overview (2)

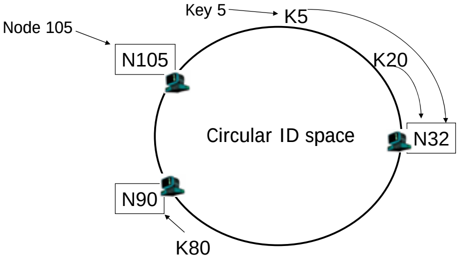
- Structured Overlay Routing:
  - **Join**: On startup, contact a “bootstrap” node and integrate yourself into the distributed data structure; get a *node id*
  - **Publish**: Route publication for *file id* toward a close *node id* along the data structure
  - **Search**: Route a query for file id toward a close node id. Data structure guarantees that query will meet the publication.
  - **Fetch**: Two options:
    - Publication contains actual file => fetch from where query stops
    - Publication says “I have file X” => query tells you 128.2.1.3 has X, use IP routing to get X from 128.2.1.3

DHT: Example - Chord

- Associate to each node and file a unique *id* in an *uni*-dimensional space (a Ring)
  - E.g., pick from the range [0...2^m]
  - Usually the hash of the file or IP address
- Properties:
  - Routing table size is O(log *N*) , where *N* is the total number of nodes
  - Guarantees that a file is found in O(log *N*) hops

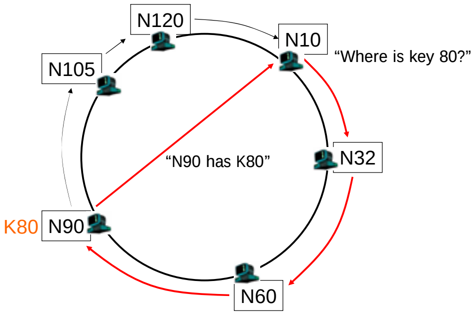
from MIT in 2001

DHT: Consistent Hashing

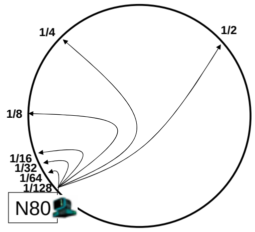


A key is stored at its successor: node with next higher ID

DHT: Chord Basic Lookup



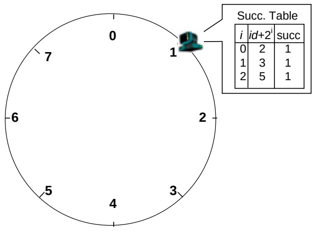
DHT: Chord "Finger Table"



- Entry *i* in the finger table of node *n* is the first node that succeeds or equals *n* + 2^i
- In other words, the ith finger points $1/2^{n-i}$ way around the ring

DHT: Chord Join

- Assume an identifier space [0..8]
- Node n1 joins

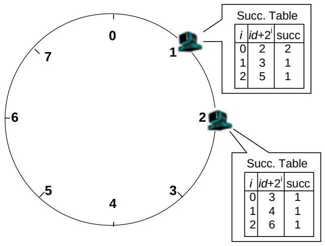


Succ. Table

i	*id*+2^i	succ
0	2	1
1	3	1
2	5	1

DHT: Chord Join

- Node n2 joins



Succ. Table

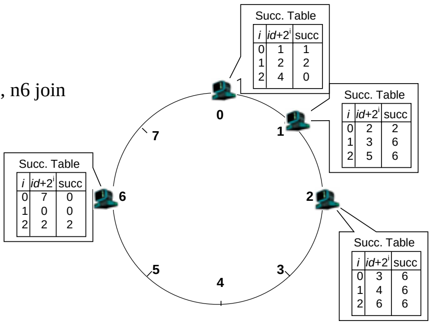
i	*id*+2^i	succ
0	2	2
1	3	1
2	5	1

Succ. Table

i	*id*+2^i	succ
0	3	1
1	4	1
2	6	1

DHT: Chord Join

- Nodes n0, n6 join



Succ. Table

i	*id*+2^i	succ
0	1	1
1	2	2
2	4	0

Succ. Table

i	*id*+2^i	succ
0	2	2
1	3	6
2	5	6

Succ. Table

i	*id*+2^i	succ
0	3	6
1	4	6
2	6	6

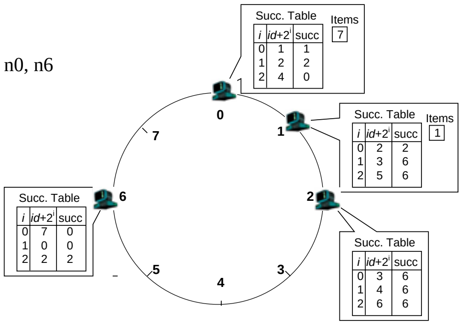
Succ. Table

i	*id*+2^i	succ
0	7	0
1	0	0
2	2	2

DHT: Chord Join

- Nodes:
  n1, n2, n0, n6

- Items:
  f7, f2



Succ. Table

i	*id*+2^i	succ
0	1	1
1	2	2
2	4	0

Items 7

Succ. Table

i	*id*+2^i	succ
0	2	2
1	3	6
2	5	6

Items 1

Succ. Table

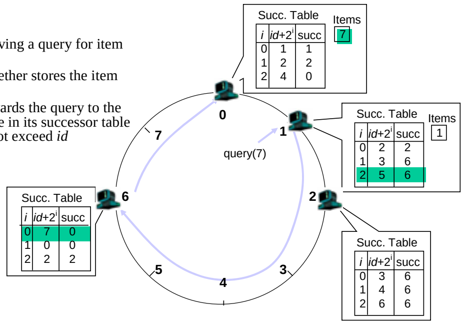
i	*id*+2^i	succ
0	3	6
1	4	6
2	6	6

Succ. Table

i	*id*+2^i	succ
0	7	0
1	0	0
2	2	2

DHT: Chord Routing

- Upon receiving a query for item *id*, a node:
- Checks whether stores the item locally
- If not, forwards the query to the largest node in its successor table that does not exceed *id*



Succ. Table

i	*id*+2^i	succ
0	1	1
1	2	2
2	4	0

Items 7

Succ. Table

i	*id*+2^i	succ
0	2	2
1	3	6
2	5	6

Items 1

Succ. Table

i	*id*+2^i	succ
0	3	6
1	4	6
2	6	6

Succ. Table

i	*id*+2^i	succ
0	7	0
1	0	0
2	2	2

DHT: Chord Summary

- Routing table size?
  - Log *N* fingers
- Routing time?
  - Each hop expects to 1/2 the distance to the desired id => expect O(log *N*) hops.

DHT: Discussion

- Pros:
  - Guaranteed Lookup
  - O(log *N*) per node state and search scope
- Cons:
  - No one uses them? (only one file sharing app)
  - Supporting non-exact match search is hard