

Introduction to Distributed Hash Tables

Eric Rescorla

Network Resonance

`ekr@networkresonance.com`

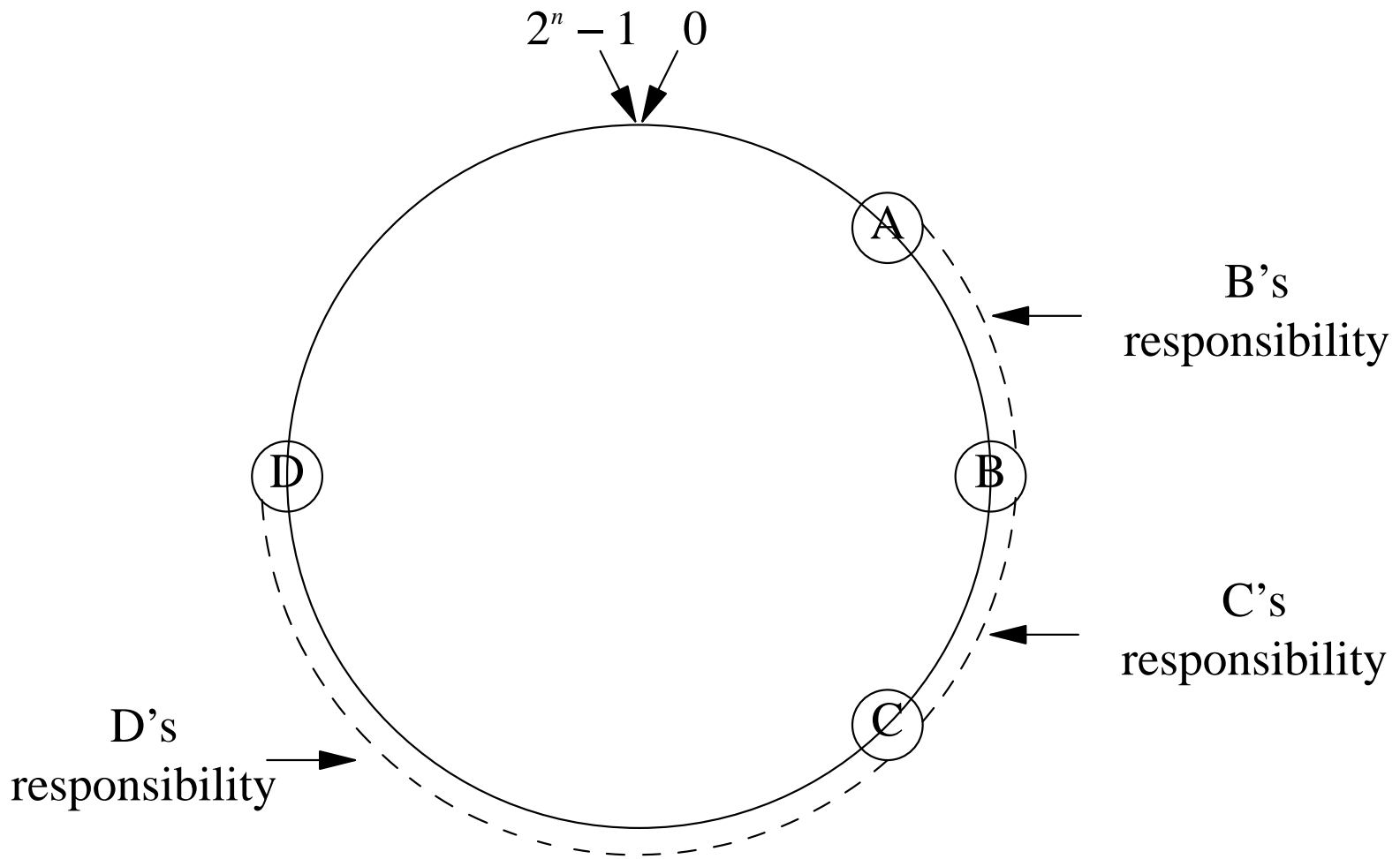
Overall Concept

- Distributed Hash Table (DHT)
- Distribute data over a large P2P network
 - Quickly find any given item
 - Can also distribute responsibility for data storage
- What's stored is key/value pairs
 - The key value controls which node(s) stores the value
 - Each node is responsible for some section of the space
- Basic operations
 - *Store(key, val)*
 - *val = Retrieve(key)*

The standard example: Chord [SMK⁺01]

- Each node chooses a n -bit ID
 - Intention is that they be random
 - Though probably a hash of some fixed info
 - IDs are arranged in a ring
- Each lookup key is also a n -bit ID
 - I.e., the hash of the real lookup key
 - Node IDs and keys occupy the same space!
- Each node is responsible for storing keys “near” its ID
 - Traditionally between it and the previous node
 - * Item is stored at “successor”
 - * Can be replicated at multiple successors

The Chord Ring



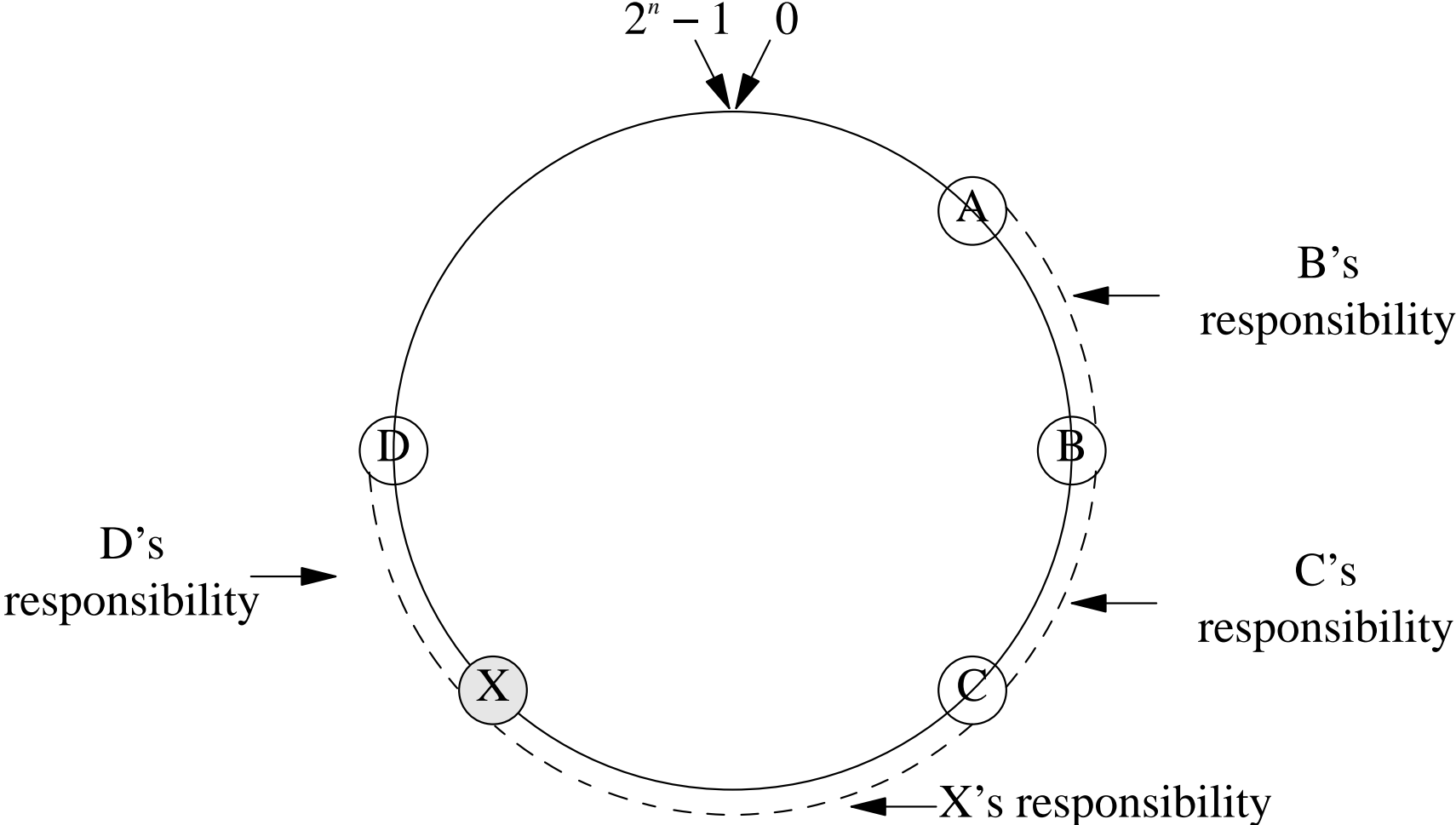
Routing

- Naive routing algorithm
 - Each node knows its neighbors
 - * Send message to nearest neighbor
 - * Hop-by-hop from there
 - Obviously this is $O(n)$
 - * So no good
- Better algorithm: “finger table”
 - Memorize locations of other nodes in the ring
 - * $a, a + 2, a + 4, a + 8, a + 16, \dots a + 2^n - 1$
 - Send message to closest node to destination
 - * Hop-by-hop again
 - * This is $\log(n)$

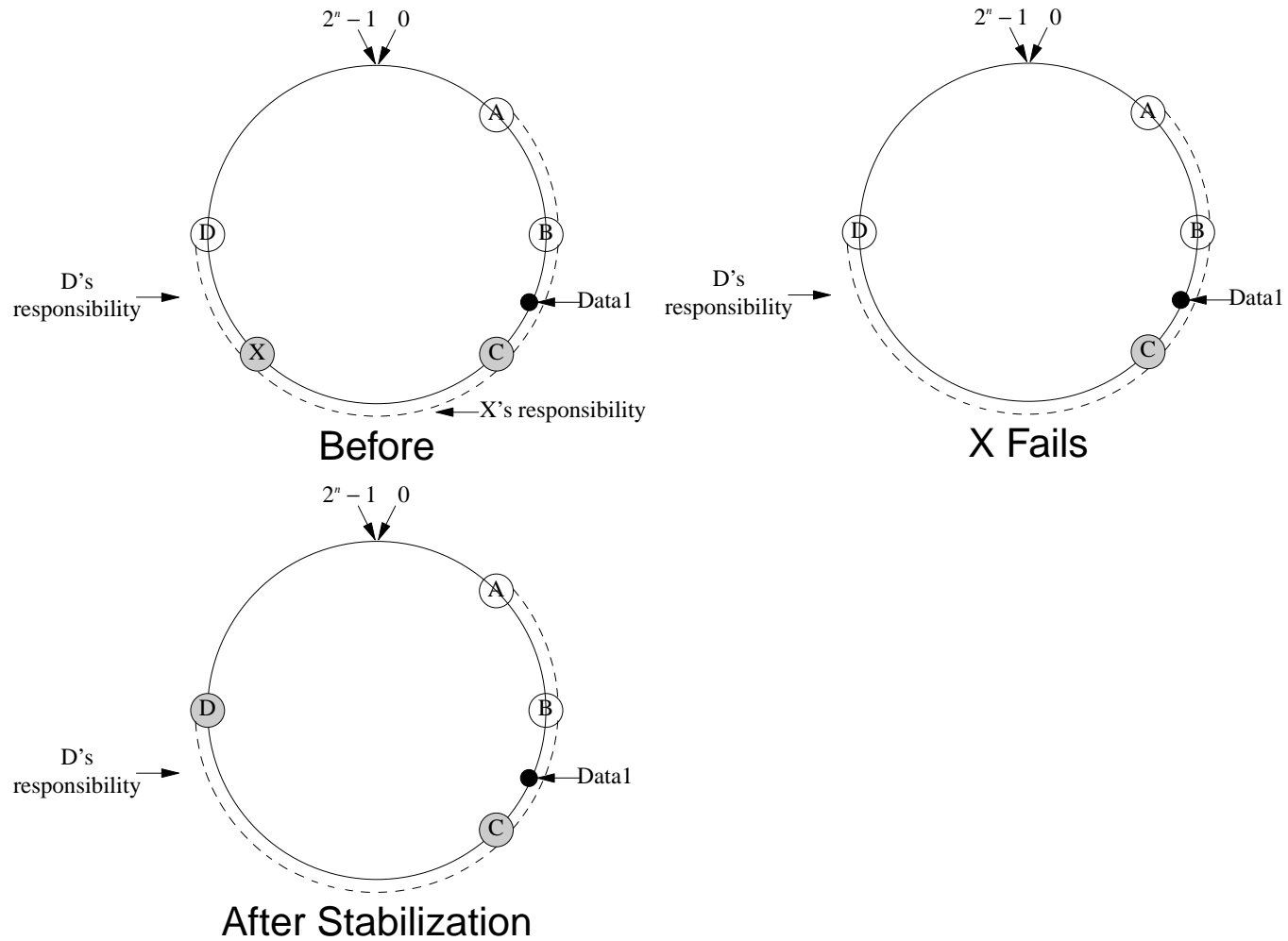
Joining

- Select a node-ID
- Contact the node that immediately follows you
 - Note that this is the same node with responsibility for your node-ID
 - Copy his state
- Data is now split up between you and the previous successor node
- Note: this requires knowing some “bootstrap node” a priori

Adding a node



Node Failure



Data must be replicated to survive node failure.

Other Structured P2P Systems

- CAN [RFH⁺01]
- Pastry [RD01]
- Tapestry [ZHS⁺01]
- Kademlia [MM02]
- Bamboo [RGRK]
- ...
- Same concept but different structure, routing algorithms, and performance characteristics

What DHTs are good at

- Distributed storage of things with known names
- Highly scalable
 - Automatically distributes load to new nodes
- Robust against node failure
 - ...except for bootstrap nodes
 - Data automatically migrated away from failed nodes
- Self organizing
 - No need for a central server

What DHTs are bad at

- Searching
 - Consequence of hash algorithm
 - “abc” and “abcd” are at totally different nodes
 - **Warning:** DHT people call lookup “search”
- Security problems
 - Hard to verify data integrity
 - Secure routing is an open problem

Example Application: Fully Distributed Name Service

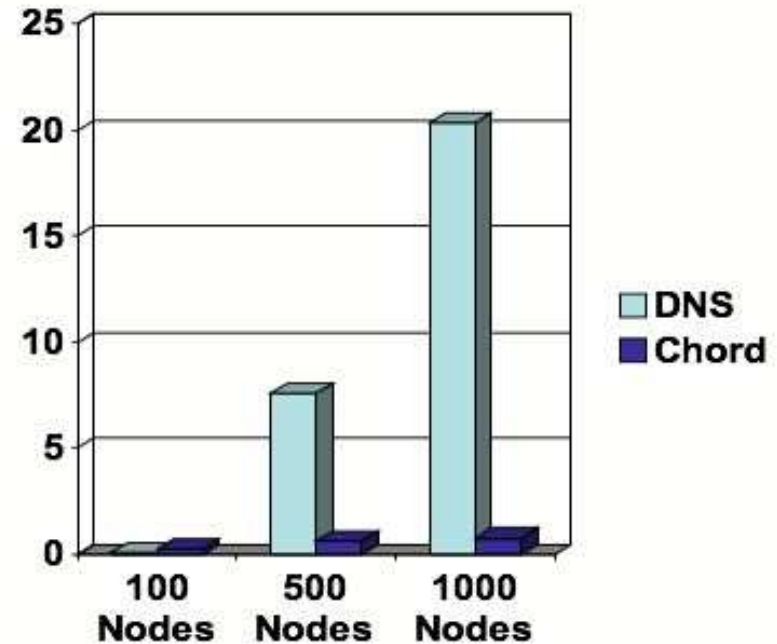
- DNS is distributed but hierarchical
 - Dependency on the roots
 - Potential single point of failure
 - No real load balancing
 - * Arguable whether this is desirable (economics)
- Can we use a DHT here?

DDNS [CMM02] and CoDoNS [RS04]

- Obvious approach: Each DNS name becomes a DHT entry
 - e.g., `www.example.com:A` → `192.0.2.7`
 - * (Just a conceptual example)
- DDNS
 - Based on Chord
 - Inferior performance to DNS ($\log(N)$ lookup cost)
- CoDoNS
 - Based on Beehive
 - $O(1)$ performance due to aggressive replication
 - * Probably unrealistic memory requirements on each node
- Both use DNSSEC for security

Performance Under Attack

- DNS
 - Attack on root nodes
- Chord
 - Attack on a continuous subspace



Percent failed queries

Data/Figure from Pappas et al. [PMTZ06]

Performance: Path Length

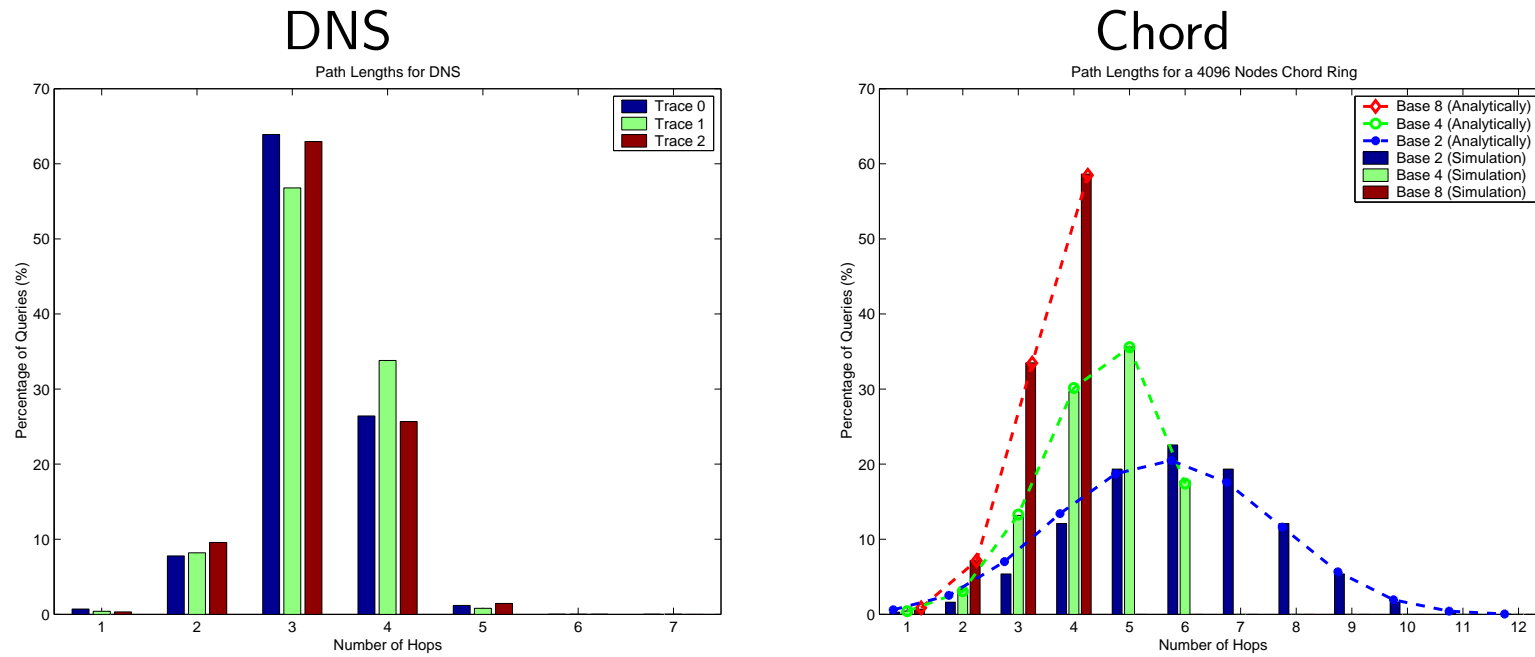


Figure from Pappas et al. [PMTZ06]

Example Application: Peer-to-Peer VoIP

- Skype Envy
- Reduce network operational costs
- Avoid having (paying) a service provider
- VoIP when there's no Internet connectivity
- Scalability
- Anonymous Calling

What's the problem?

- SIP is *already* mostly P2P
- SIP UAs can already connect directly to each other
 - But in practice they go through a centralized server
 - Modulo firewall and NAT traversal issues
- The problem is locating the right peer to connect to
 - Currently this is done with DNS
 - * Works fine with stable centralized servers
 - But how do you lookup the location of unstable peers?
 - What about dynamic DNS?
 - * Concerns about performance
 - * What if you're disconnected from the Internet?

draft-bryan-sipping-p2p-02 [BLJ06]

- Uses a DHT for location
 - Specified for Chord
 - ... but could be anything
- REGISTER by storing your location in DHT
 - Under your URL
- Calling node looks up your URL in the DHT
 - ... and connects
- This is a strawman design
 - Not even a WG yet (BOF yesterday, ad hoc tomorrow)
 - Known security problem

Overview of Security Issues

- Data correctness
- Correctness of routing
- Fairness and detecting defection
- DoS

Data Correctness

- Storing nodes have no relationship to data owner
- What stops me from overwriting data?
 - Nothing!
- And how do I know it's right when I get it?
- General approach: make sure data is verifiable
 - Self-certifying (e.g., $k = SHA1(data)$)
 - Externally signed

A simple attack: chosen Node-ID

- Assume you want to impersonate a specific value k
 - Generate a node between k and $successor(k)$
 - You're now $successor(k)$
- General fix: make it hard for people to choose their own Node-Id freely
 - Chord uses $SHA1(IPaddress)$
 - This isn't perfect
 - * An attacker who controls a big IP address space can generate a lot of IDs until it finds one it likes
 - * IPv6 makes this situation much worse

Node impersonation

- Why bother with choosing your Node-Id
 - Just impersonate the current *successor*(*k*)
 - This requires subverting Internet routing
- One natural defense: public key cryptography
 - $NodeId = SHA1(PublicKey)$
 - Easy for peers to verify
 - But this makes it easy to generate chosen NodeIDs by trial and error
 - Can use a CGA variant here: $H(IP) || H(PublicKey)$

Sybil Attacks

- What if you had a lot of bad nodes
 - Just register with the DHT a lot of times
 - Interfere with most or all routing
 - For any lookup key
- Potential defenses
 - Proof-of-work for registration
 - * Usual concerns about variance in machine performance
 - Reverse Turing Tests – but who would administer them
 - Certified Node-IDs
 - * Requires a central authority

Routing Attacks and Defenses

- General concept: get all stored replicas with high probability
- Current state of the art [CDG⁺02]
 - Failure test
 - * Detect density of replica set
 - * Compare to own neighbor set density
 - * Fake replica sets should be less dense
 - Redundant routing
 - * Only used when routing failure detected
 - * Expensive but high probability of success
- Assumes secure NodeID assignment
- Even more complicated with topology-based routing [CKS⁺06]

Fairness

- File storing costs resources
- How do you make sure people do their fair share?
- Basically an unsolved problem
 - Auditing
 - Cheating detection?

DoS

- Not much work done here
- Often possible to force system into pathological thrashing-type behavior
- Even worse if you compromise or attack a bootstrap node
- How do you do cost containment?
 - Make other people store a lot of data for you
- Force expensive secure routing algorithms

Summary

- A technically sweet technology
- Some obvious applications
- Still under very active research
- Some unsolved security problems
- Need to make sure capabilities match applications

References

- [ATS] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*.
- [BLJ06] David Bryan, Bruce Lowekamp, and Cullen Jennings. A P2P Approach to SIP Registration and Resource Location. draft-bryan-sipping-p2p-02, March 2006.
- [BS04] Salman Baset and Henning Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. September 2004.
- [CDG⁺02] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. *In the Proceedings of OSDI, 2002*.
- [CKS⁺06] Tyson Condie, Varun Kacholia, Sriram Sank, Joseph M. Hellerstein, and Petros Maniatis. Churn as Shelter. *Proceedings of the 13th Annual Network and Distributed Systems Symposium, February 2006*.

- [CMM02] Russ Cox, Athicha Muthitacharoen, and Robert T. Morris. Serving DNS using a Peer-to-Peer Lookup Service. *Proceedings of the 1st Workshop on Peer-to-Peer Systems (IPTPS)*, Cambridge, MA, 2002.
- [DKK⁺01] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with CFS. *Proceedings of ACM SOSP 2001*, October 2001.
- [MM02] Petar Maymounkov and David Mazires. Kademia: A Peer-to-peer Information System Based on the XOR Metric. *1st International Workshop on Peer-to-peer Systems*, March 2002.
- [PMTZ06] Vasileios Pappas, Daniel Massey, Andreas Terzis, and Lixia Zhang. A Comparative Study of Current DNS with DHT-Based Alternatives. April 2006.
- [RD01] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. *In the Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, November 2001.
- [RFH⁺01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. *Proc. of the*

conf. on Applications, technologies, architectures and protocols for computer communications, August 2001.

- [RGRK] Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatoicz. Handling Churn in a DHT. *Proceedings of the USENIX Annual Technical Conference*.
- [RM06] John Risson and Tim Moors. Survey of research towards robust peer-to-peer networks: Search methods. draft-irtf-p2prg-survey-search-00.txt, March 2006.
- [RS04] Venugopalan Ramasubramanian and Emin Gn Sirer. The Design and Implementation of a Next Generation Name Service for the Internet. *In the Proceedings of ACM SIGCOMM*, 2004.
- [SMK⁺01] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. *In the Proceedings of ACM SIGCOMM*, August 2001.
- [ZHS⁺01] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John D. Kubiatoicz. Tapestry: A Resilient Global-Scale

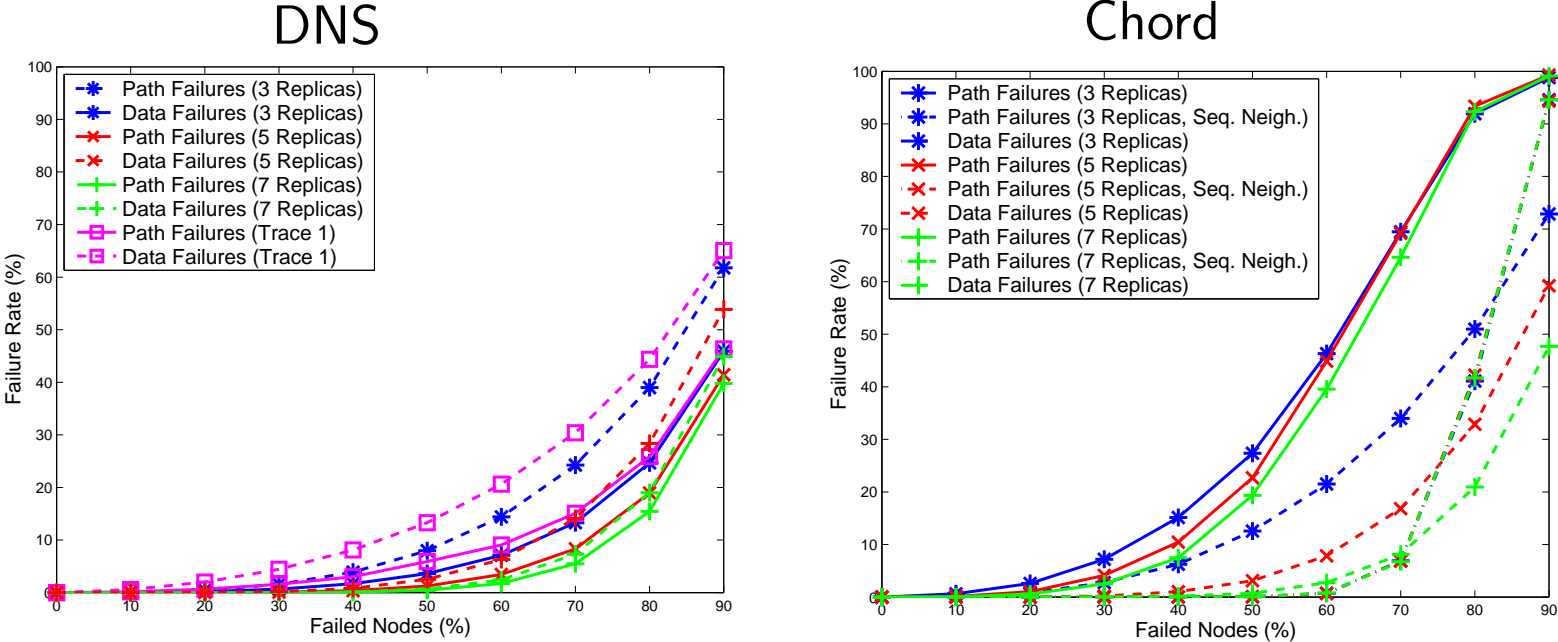
Overlay for Service Deployment. *In the Proceedings of IEEE Journal on Selected Areas in Communications*, 22(1), January 2001.

BACKUP SLIDES

Ring Stabilization

- Need to propagate joins and leaves
- Periodically ask your successor who his predecessor is
 - If it's after you then it's your successor
 - Notify it and update yourself
 - rinse, repeat
- Ring works even if not completely consistent
 - Performance just isn't as good

Availability - Random Failures



Example Application: Distributed File Storage

- Why a distributed file system?
 - Anywhere access to information
 - File sharing
 - * Especially multimedia files
- Naive design
 - Store each file at node(s) corresponding to its name
 - Bad load balancing
 - * Some files are more popular than others
 - * Unlucky servers get hammered
 - Name collisions
 - * Who has the right to store “Crossroads”?
 - And which version is it?

Solving the name collision problem

- Don't use user-friendly names
 - We've just established that they're overloaded anyway
- Use Hash(file) as lookup key
 - This guarantees uniqueness
 - * At least statistically
 - Plus you can verify correctness
 - * Just recompute the hash and compare to lookup key

Cooperative File System [DKK⁺01]

- Based on Chord and DHash
- Store blocks instead of files
 - Automatically provides load balancing
 - Any substantial file will be split across many servers
 - * “Virtual servers” allow even better load balancing
 - Blocks are cached along their Chord lookup path
 - * Provides offloading for popular files
- Each block stored under its hash value
 - “Root-block” contains pointers to file blocks
 - Root block is signed
 - * Stored under $Hash(PublicKey)$
- Note: this does not solve the directory problem

Real P2P Systems Let You Search

- “Give me every song from Blonde on Blonde”
 - “Dylan, Bob” or “Bob Dylan”? How do you spell “Blonde”?
- SHA-1 of “Dylan, Bob”, “Bob Dylan”, and “Dylan” are all unrelated
 - And stored on totally different nodes. Try all variants????
 - And what about free text search?
- Successful P2P file sharing systems allow search
 - Centralized: Napster, Torrent trackers, etc.
 - Decentralized: flooding
- DHTs offer no leverage here
 - You could build an index on the DHT [BKKMS03]
 - But not particularly efficient [LHSH03]

Background: Skype

- P2P Voice Application
 - 241.5 Million Downloads
 - Millions online at once
 - 1.9B Minutes Served
- Advertised as p2p VoIP, but?
 - Supernodes
 - Centralized Login Server
 - Namespace ownership
 - Hands out certs signed by Skype
 - SIP-based PSTN Interconnect server
 - All encrypted and all proprietary

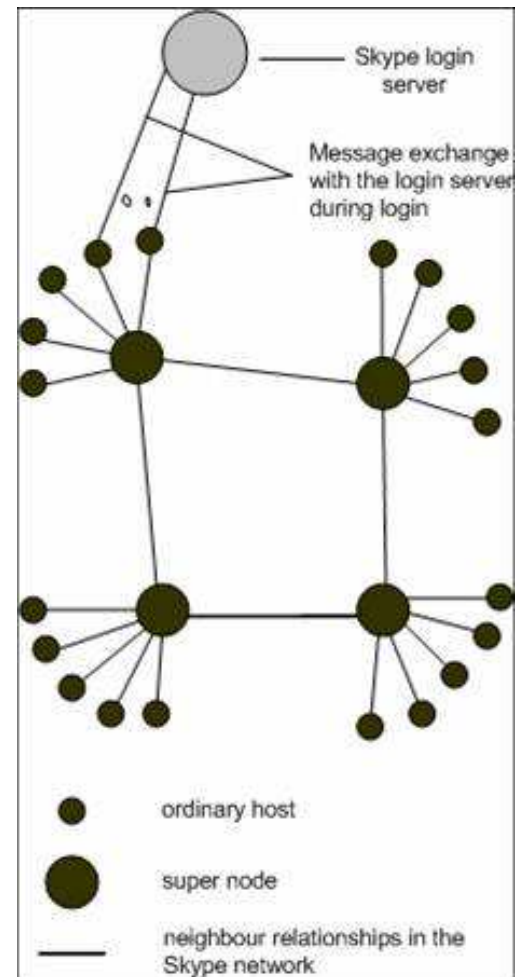


Diagram from Baset and Schulzrinne [BS04]