Handling Churn in a DHT

USENIX Annual Technical Conference
June 29, 2004

Sean Rhea, Dennis Geels,
Timothy Roscoe, and John Kubiatowicz

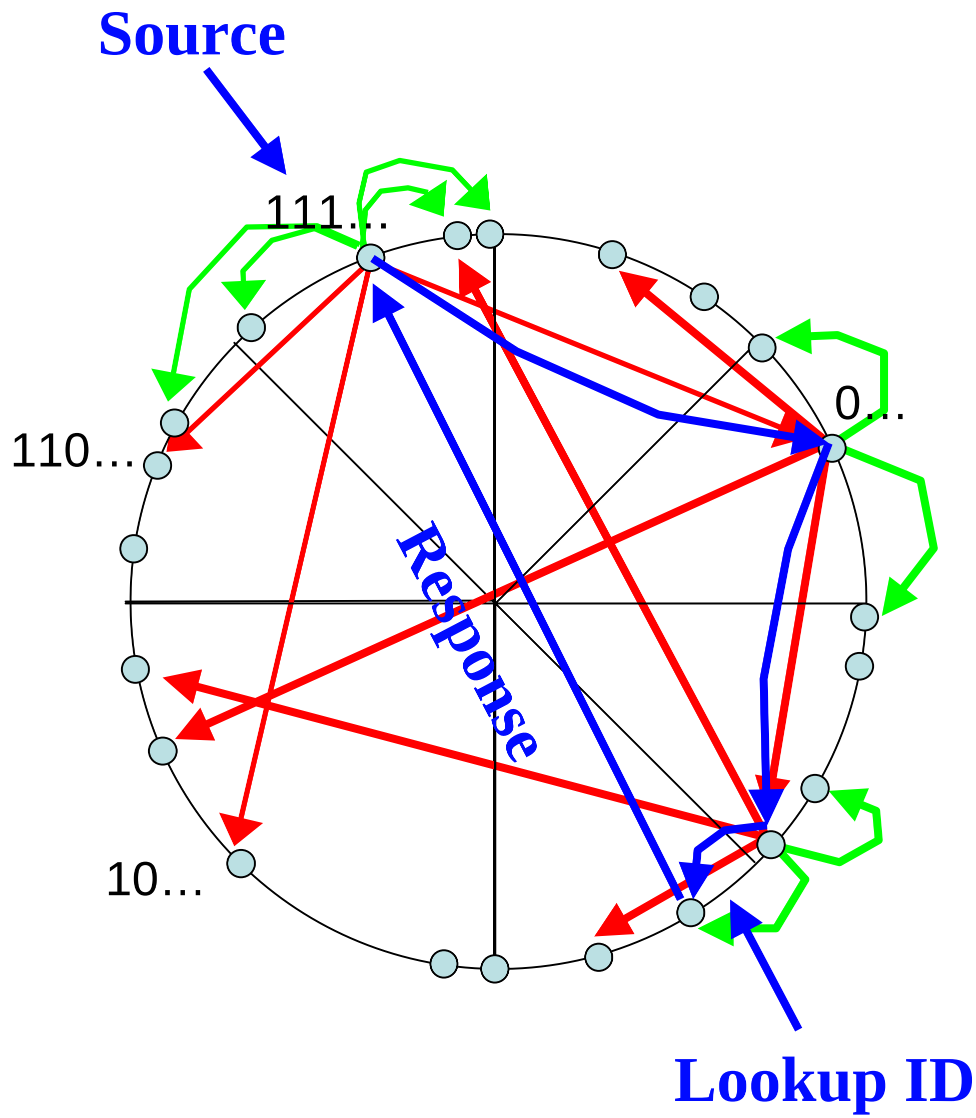
UC Berkeley and Intel Research Berkeley

What’s a DHT?

- Distributed Hash Table
  - Peer-to-peer algorithm to offering put/get interface
  - Associative map for peer-to-peer applications
- More generally, provide *lookup* functionality
  - Map application-provided hash values to nodes
  - (Just as local hash tables map hashes to memory locs.)
  - Put/get then constructed above lookup
- Many proposed applications
  - File sharing, end-system multicast, aggregation trees

How Does Lookup Work?

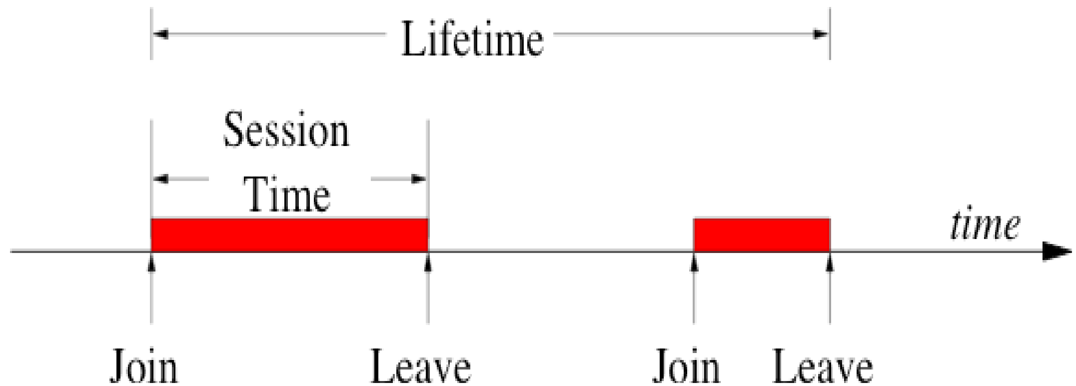
- Assign IDs to nodes
  - Map hash values to node with closest ID
- Leaf set is successors and predecessors
  - All that's needed for correctness
- Routing table matches successively longer prefixes
  - Allows efficient lookups



Why Focus on Churn?

Chord is a "scalable protocol for lookup in a dynamic peer-to-peer system with frequent node arrivals and departures"

-- Stoica et al., 2001



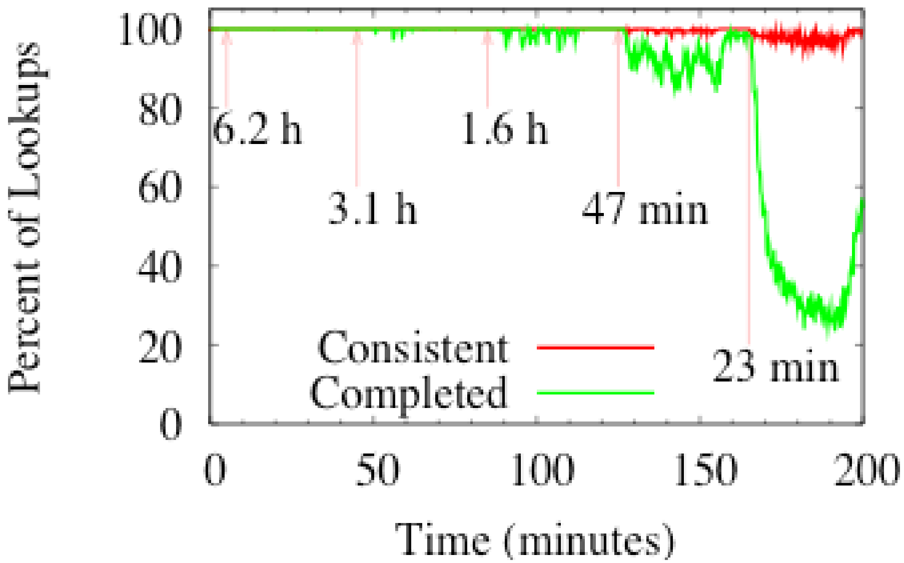
Authors	Systems Observed	Session Time
SGG02	Gnutella, Napster	50% < 60 minutes
CLL02	Gnutella, Napster	31% < 10 minutes
SW02	FastTrack	50% < 1 minute
BSV03	Overnet	50% < 60 minutes
GDS03	Kazaa	50% < 2.4 minutes

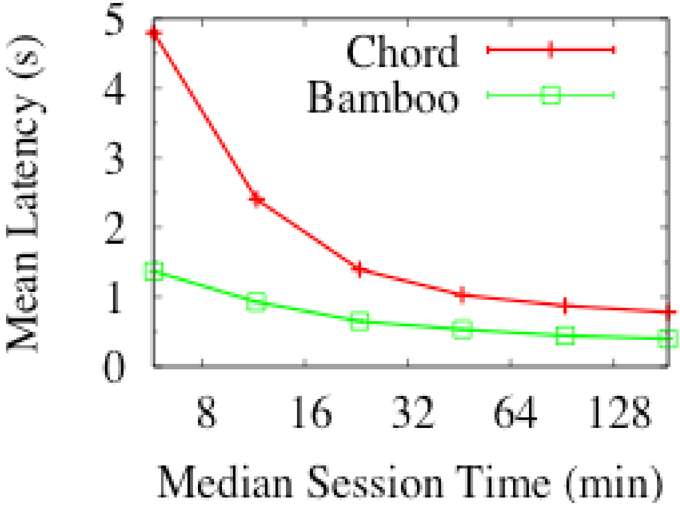
A Simple *lookup* Test

- Start up 1,000 DHT nodes on ModelNet network
  - Emulates a 10,000-node, AS-level topology
  - Unlike simulations, models cross traffic and packet loss
  - Unlike PlanetLab, gives reproducible results
- Churn nodes at some rate
  - Poisson arrival of new nodes
  - Random node departs on every new arrival
  - Exponentially distributed session times
- Each node does 1 lookup every 10 seconds
  - Log results, process them after test

Early Test Results

- Tapestry (the OceanStore DHT) falls over completely
  - Worked great in simulations, but not on more realistic network
  - Despite sharing almost all code between the two
- And the problem isn't limited to Tapestry:





Handling Churn in a DHT

- Forget about comparing different impls.
  - Too many differing factors
  - Hard to isolate effects of any one feature
- Implement all relevant features in one DHT
  - Using Bamboo (similar to Pastry)
- Isolate important issues in handling churn
  1. Recovering from failures
  2. Routing around suspected failures
  3. Proximity neighbor selection

Recovering From Failures

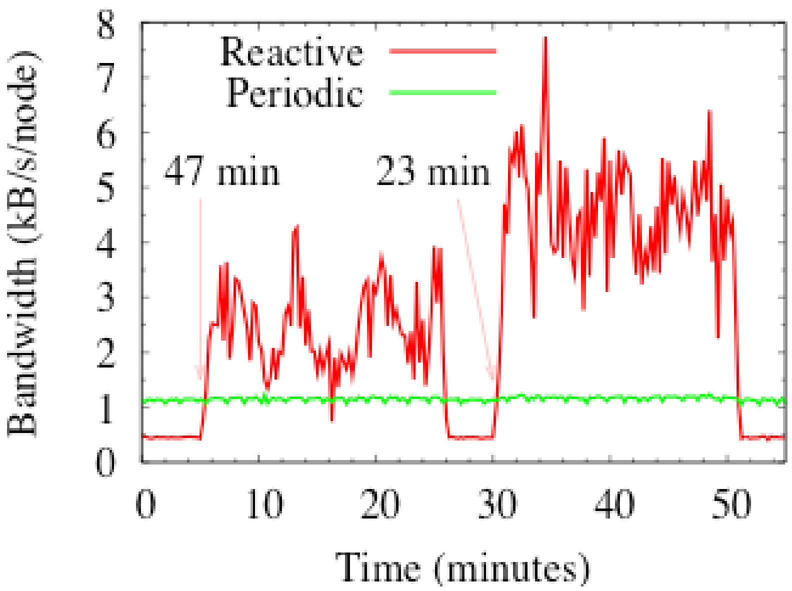
- For correctness, maintain leaf set during churn
  - Also routing table, but not needed for correctness
- The Basics
  - Ping new nodes before adding them
  - Periodically ping neighbors
  - Remove nodes that don’t respond
- Simple algorithm
  - After every change in leaf set, send to all neighbors
  - Called *reactive* recovery

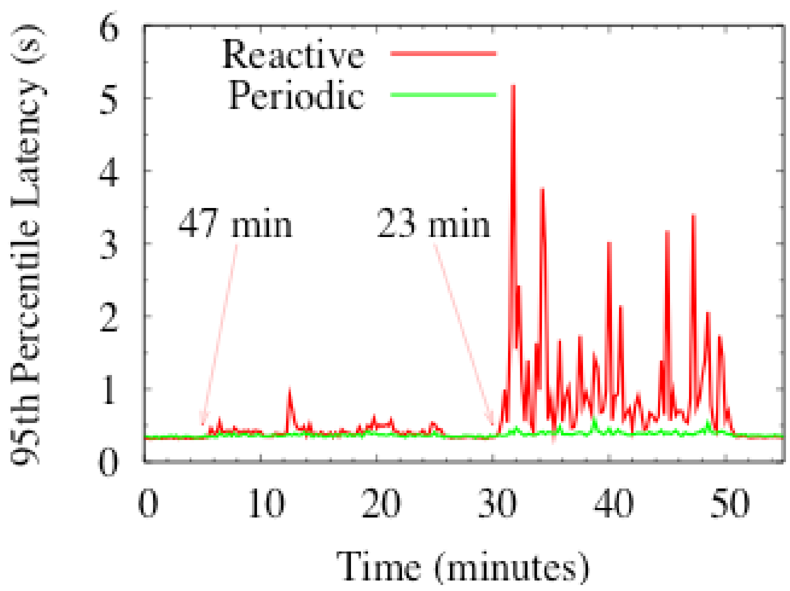
The Problem With Reactive Recovery

- Under churn, many pings and change messages
  - If bandwidth limited, interfere with each other
  - Lots of dropped pings looks like a failure
- Respond to failure by sending more messages
  - Probability of drop goes up
  - We have a positive feedback cycle (squelch)
- Can break cycle two ways
  1. Limit probability of “false suspicions of failure”
  2. Recovery periodically

Periodic Recovery

- Periodically send whole leaf set to a random member
  - Breaks feedback loop
  - Converges in O(log N)
- Back off period on message loss
  - Makes a negative feedback cycle (damping)



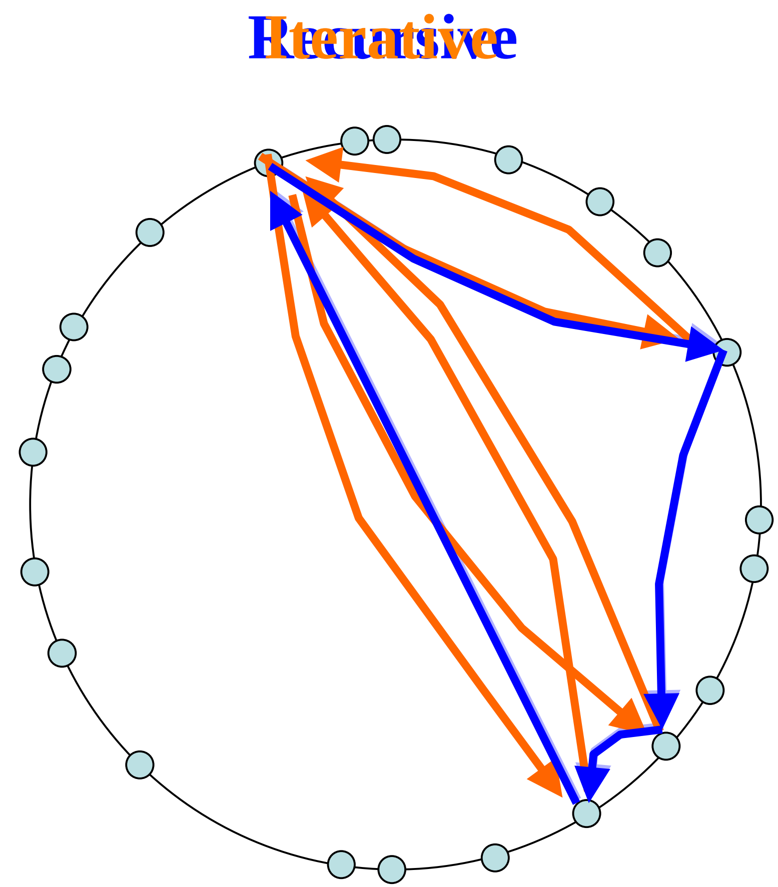


Routing Around Failures

- Being conservative increases latency
  - Original next hop may have left network forever
  - Don’t want to stall lookups
- DHT has many possible routes
  - But retrying too soon leads to packet explosion
- Goal:
  1. Know for sure that packet is lost
  2. Then resend along different path

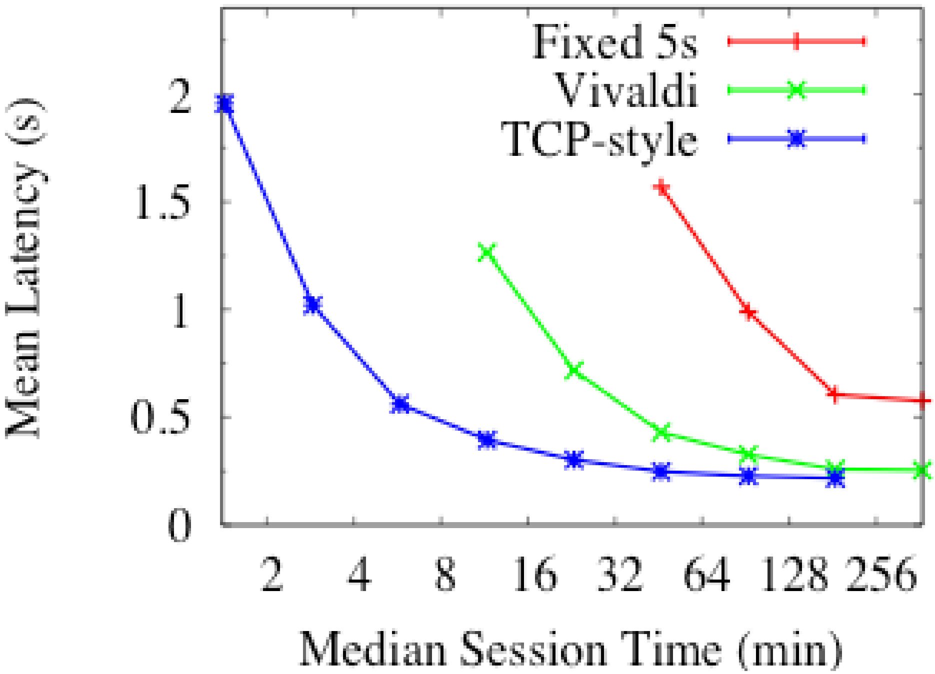
Calculating Good Timeouts

- Use TCP-style timers
  - Keep past history of latencies
  - Use this to compute timeouts for new requests
- Works fine for *recursive* lookups
  - Only talk to neighbors, so history small, current
- In *iterative* lookups, source directs entire lookup
  - Must potentially have good timeout for *any* node



Virtual Coordinates

- Machine learning algorithm to estimate latencies
  - Distance between coords. proportional to latency
  - Called Vivaldi; used by MIT Chord implementation
- Compare with TCP-style under recursive routing
  - Insight into cost of iterative routing due to timeouts



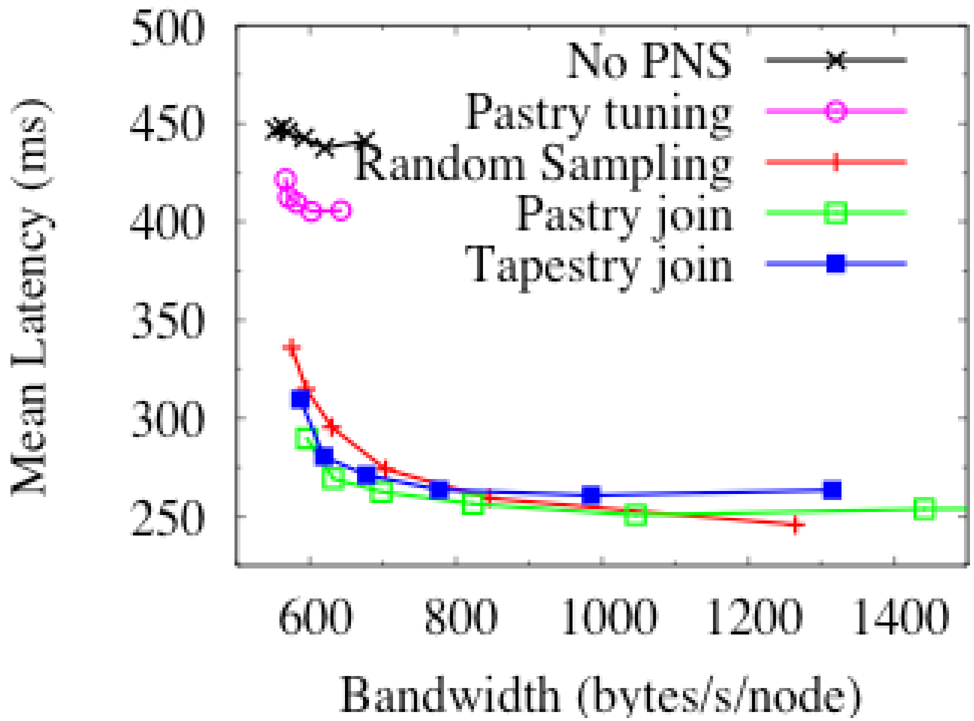
Proximity Neighbor Selection (PNS)

- For each neighbor, may be many candidates
  - Choosing closest with right prefix called PNS
  - One of the most researched areas in DHTs
  - Can we achieve good PNS under churn?
- Remember:
  - leaf set for correctness
  - routing table for efficiency?
- Insight: extend this philosophy
  - Any routing table gives O(log N) lookup hops
  - Treat PNS as an optimization only
  - Find close neighbors by simple random sampling

PNS Results

(very abbreviated--see paper for more)

- Random sampling almost as good as everything else
  - 24% latency improvement free
  - 42% improvement for 40% more b.w.
  - Compare to 68%-84% improvement by using good timeouts
- Other algorithms more complicated, not much better



Related Work

- Liben-Nowell et al.
  - Analytical lower bound on maintenance costs
- Mahajan et al.
  - Simulation-based study of Pastry under churn
  - Automatic tuning of maintenance rate
  - Suggest increasing rate on failures!
- Other simulations
  - Li et al.
  - Lam and Liu
- Zhuang
  - Cooperative failure detection in DHTs
- Dabek et al.
  - Throughput and latency improvements w/o churn

Future Work

- Continue study of iterative routing
  - Have shown virtual coordinates good for timeouts
  - How does congestion control work under churn?
- Broaden methodology
  - Better network and churn models
- Move beyond lookup layer
  - Study put/get and multicast algorithms under churn

Conclusions/Recommendations

- Avoid positive feedback cycles in recovery
  - Beware of “false suspicions of failure”
  - Recover periodically rather than reactively
- Route around potential failures early
  - Don’t wait to conclude definite failure
  - TCP-style timeouts quickest for recursive routing
  - Virtual-coordinate-based timeouts not prohibitive
- PNS can be cheap and effective
  - Only need simple random sampling

For code and more information:
bamboo-dht.org