Examination of Routing Algorithms in Distributed Hash Tables (DHTs) for Peer-to-Peer (P2P) Networks

Kevin Thomas

http://www.sfu.ca/kta18/ENSC835Project.html
kta18@sfu.ca
Presentation Outline

Introduction

- What are P2P networks and DHTs?
- DHT Specifics: Addressing, Overlays, Routing and Churn
- Project Scope

Project Implementation Details

- Diagrams
- Simulation Tool(s) and Methodology
- Simulation Runs and Results

Conclusion

- Difficulties Experienced, Lessons Learned
- Current and Future Work
- Real-life Applications of DHTs

References
Peer-to-Peer (P2P) Networks & Distributed Hash Tables (DHTs)

✎ A P2P network/system is a “self-organizing system of equal, autonomous entities (peers) which aims for the shared usage of distributed resources in a networked environment avoiding central services” [1].

✎ Implementation challenges of massively distributed P2P systems

✎ Lookup problem – finding other nodes in an efficient manner.

✎ DHTs give structure to massive P2P systems

✎ Guarantee location and retrieval of any kind of data distributed across nodes.
DHT Specifics: Addressing, Overlays, Routing and Churn

- DHTs make use of a distributed hash function running on each node to map nodes and data items into a common 'virtual' address space, independent of network topology. An example of a hash function is Secure Hash Algorithm 1 (SHA1) [2].

- In this manner, an 'overlay' network is created that essentially runs on top of the existing network topology.

- Routing is achieved by having each node maintain a table of limited no. of references to addresses of successors, predecessors and neighbors within the address space. Locating a node and data item should take approximately $O(\log N)$ hops. Routing can be recursive or iterative [3].

- Churn: a term for node arrival and departure/failure within the DHT.
Project Scope

⇒ Many interesting approaches to DHT implementations exist: Chord, Pastry, Tapestry, CAN, Kademlia, Viceroy, Symphony, Broose, Koorde, Gia, etc. Pick one, implement and analyze.

⇒ For this project, the Chord algorithm was used.

⇒ Examine characteristics of the implemented DHT under operation, with and without churn and using iterative and recursive lookup procedures.

⇒ Understand issues to be dealt with when implementing DHTs
Chord uses $l$-bit identifiers, integers in the range $[0, 2^l - 1]$ as keys to map node and data keys to a one-dimensional circular virtual address space.
Simulation Tool(s) and Methodology

- OMNeT++ Discrete Event Simulator [4]
- INET Framework for OMNet++ [5]
- OverSim: P2P Overlay Simulation Framework for OMNeT++ [6, 7]
- Four simulation groups using Chord DHT, with 16, 32, 64 and 128 nodes:
  - Recursive lookup using a simple network
  - Iterative lookup using a simple network
  - Iterative lookup using an IPv4 network
  - Recursive lookup using a simple network and faster stabilization
- All tools and simulations ran on an Ubuntu Linux workstation
Simulation Runs and Results

• Screenshot of the OverSim simulation modules for Run 3, using an IPv4 underlay.
Simulation Runs and Results

🌠 A screenshot of the Chord simulation using 9 overlay nodes:
Simulation Runs and Results

➲ Run 1 (Chord Recursive, Simple Network, 1 hour simulation with 16 and 32 peers)

➲ Plot of Current Delivery Ratio (Percentage of successfully delivered messages) vs. Time (s)
Run 1 (Chord Recursive, Simple Network, 1 hour simulation with 64 and 128 peers)

Plot of Current Delivery Ratio (Percentage of successfully delivered messages) vs. Time (s)
Run 2 (Chord Iterative, Simple Network, 1 hour simulation with 16 and 32 peers)

Plot of Current Delivery Ratio (Percentage of successfully delivered messages) vs. Time (s)
Run 2 (Chord Iterative, Simple Network, 1 hour simulation with 64 and 128 peers)

Plot of Current Delivery Ratio (Percentage of successfully delivered messages) vs. Time (s)
Run 3 (Chord Iterative, IPv4 Network, 1 hour simulation with 16 and 32 peers)

Plot of Current Delivery Ratio (Percentage of successfully delivered messages) vs. Time (s)
Run 3 (Chord Iterative, IPv4 Network, 1 hour simulation with 64 and 128 peers)

Plot of Current Delivery Ratio (Percentage of successfully delivered messages) vs. Time (s)
Run 4  (Chord Recursive, Simple Network, faster stabilization, 1 hour simulation with 16 and 32 peers)

Plot of Current Delivery Ratio (Percentage of successfully delivered messages) vs. Time (s)
Run 4 (Chord Recursive, Simple Network, faster stabilization, 1 hour simulation with 64 peers)

Plot of Current Delivery Ratio (Percentage of successfully delivered messages) vs. Time (s)
Conclusion

Difficulties Experienced, Lessons Learned

➢ Time lost with initial attempt at using OPNET
➢ Learning curve with shift to a new simulation tool (OverSim) and unfamiliar OS (Linux)

Current and Future Work

➢ Tweaking the various parameters to obtain optimum performance
➢ Load balancing approaches within DHT implementations
➢ An adaptable DHT protocol
References


