Distributed, Secure Load Balancing with Skew, Heterogeneity, and Churn

Jonathan Ledlie and Margo Seltzer
INFOCOM 2005 - March 16, 2005
Motivation - Why balance DHTs?

• Distributed hash tables (DHTs):
  – Becoming “off-the-shelf” distributed data structures
  – Was: backup storage; now: ALM, resource discovery

• DHTs must be versatile:
  – Handle variety of loads - low msg loss
    • Allocate network capacity
  – Realistic network conditions
  – Reasonably secure

• Numerous load balancing proposals in literature
  – Unrealistic assumptions
  – Poor performance
## Problematic Assumptions

<table>
<thead>
<tr>
<th>Assumption</th>
<th>Reality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical Nodes</td>
<td>Uniform Capacity</td>
</tr>
<tr>
<td>Workload</td>
<td>Uniform</td>
</tr>
<tr>
<td>Membership</td>
<td>Stable</td>
</tr>
<tr>
<td>Security</td>
<td>Pick any ID</td>
</tr>
</tbody>
</table>

### Current load balancing algorithms are insufficient
k-Choices Algorithm

• Support variation in skew, node heterogeneity, and churn
• Make IDs verifiable

1. Sample
2. Cost fn
3. Join
Talk Outline

• Overview
• Preliminaries
  – DHTs
  – Security
  – Network Characteristics
• k-Choices
• Prior Techniques
• Evaluation
• Conclusion
DHTs - Refresher

- Each node has one or more virtual servers (VSs).
- Each virtual server has an ID namespace (e.g., \((0,1], (0,2^{160}]\)).
- Msgs via \(O(\log(N))\) hops between any two VSs.

Chord-like routing

\[(a,b,c)\]
\[(d)\]
\[(e,f)\]
\[(g,h)\]
DHTs - Load

Load

(a,b,c)
(d)
(e,f)
(g,h)
(i,j)
Sybil Attacks

Unsecured IDs→Take over portions of ring
Sybil Attack - Solution

- Central authority certifies each ID [Castro02]
- k-Choices uses similar scheme to generate limited set of certified IDs.
Outline

• Overview
• Preliminaries
  – DHTs
  – Security
  – Network Characteristics
• k-Choices
• Prior Techniques
• Evaluation
• Conclusion
Characteristics - Skew

- Skew: hotspots popular content
- Typically Zipf popularity
- E.g., Gnutella queries (log-log scale):

![Graph showing log-log scale]
Characteristics - Churn

- Churn: pattern of participant join and departure.
- Pareto (memory-full) distribution (60 minute avg).
Characteristics - Heterogeneity

- Network bandwidths vary by five orders-of-magnitude.
- Routing capacity varies widely.
Outline

• Overview
• Preliminaries
  – DHTs
  – Security
  – Network Characteristics
• k-Choices
• Prior Techniques
• Evaluation
• Conclusion
k-Choices - Steps

1. Probe
2. Evaluate Cost Function
3. Join
k-Choices - Sample

Discover load and capacity at each ID

Sample ID b:
Learn succ(b) actual load, target load, and node capacity.

Load          Capacity
           [ ]        [ ]

Over target
k-Choices - Cost Function

Choose ID that minimizes mismatch between target and load normalized by capacity.
k-Choices - Properties

- Incorporates workload skew and node heterogeneity.
- Proactive load balancing - join time
- Reactive load balancing - reselect ID
- Verifiable IDs
Outline

- Overview
- Preliminaries
- k-Choices
- Prior Techniques
  - \( \log(N) \) virtual servers
  - Transfer
  - Proportion
  - Threshold
- Evaluation
- Conclusion
Prior Work - $\log(N)$ VS

- Namespace balancing (e.g. [Karger97])
- Central Limit Theorem
  - Total namespace for each node approximately equal

Namespace balancing does not equal load balancing.

[Graph showing comparison between Random and Perfectly Balanced routing success rate with varying skew: Zipf alpha values.]
Prior Work - Transfer

- Overload:
  a) >1 VS: attempt to transfer
  b) 1 VS: split first, then transfer
- Pros: Simple, Good Performance
- Cons: Unsecure
  - Split to arbitrary ID (cut in half)
  - Transfer to anyone

[Rao03, Godfrey04]
Evaluation

• Trace Driven Simulation

• Results
  – Determining $k$
  – Vary applied load
  – Vary churn
  – Vary skew

• Pastry Implementation
  – Throughput
  – Heterogeneous real node bandwidths (Emulab)
Results - Choosing k

4k nodes, avg capacity=100 m/s, 60 min avg lifetime

k=8 sufficiently reduced utilization.
Results - Trace

5508 nodes; median capacity: 191 msgs/sec

k-Choices and Transfer performed equally well with skewed workloads.
Results - Implementation

Pastry; “lookup+download”; 64x4 nodes - last mile limited

k-Choices: 20% throughput improvement
Conclusion

• k-Choices:
  – Approx. same performance as Transfer
  – Doesn’t change security properties
  – Not the final word - range queries

• Design for empirical system
  – Namespace balancing?
  – Skew, wide capacity distribution, churn
  – Security: Sybil attacks
Questions?

• Thanks!

• Contact:
  – Jonathan Ledlie
  – jonathan@eecs.harvard.edu
Prior Work - Threshold

• If our utilization has increased beyond threshold
  – Compare utilization to neighbors
  – Shift their IDs?

• Else
  – Compare to set of log(N) random VSs
  – Move best to be our new predecessor

[Ganesan04]
Prior Work - Proportion

- Overload: shed VSs
- Underload: create them

- Pros: No communication
- Cons:
  - Large number of VSs created
  - New lowest common denominator
  - Cascading deletes

[Dabek01]