Minimizing Churn in Distributed Systems

by
P. Brighten Godfrey, Scott Shenker, and Ion Stoica

appearing in
SIGCOMM 2006

presented by
Todd Sproull

Introduction

- Problem: nodes joining or leaving distributed systems creates something called churn
  - Churn may reveal itself through
    - Dropped messages
    - Increase Latency
    - Increased Bandwidth
  - High churn rate can increase costs or decrease service quality
    - Due to maintaining consistency between nodes

- Current solutions
  - Pick nodes before start of experiment and hope they do not fail
  - Pick next node to join the set based on a particular metric
    - Uptime
    - Latency
    - Load

- Proposed approach
  - Randomly select the next node to join the set
Overview of Paper

- Provides comparison of different node selection techniques on five real-world traces
- Explains why a random approach performs so well
  » Analysis of different strategies
- Demonstrate applications using a variety of node selection techniques
  » Anycast
  » Overlay multicast
  » Distributed Hash Table (DHT) Neighbor Selection
  » DHT Data Replication

Model

- Consider n nodes in the system
- Nodes either up or down
  » Nodes in up state are either
    - Available
    - In use
- There exists some subset of target nodes
  » \( k = \alpha \times n \)
    - Where \( 0 < \alpha \leq 1 \)
  » Consider different scenarios in which we wish to use \( k \) out of \( n \geq k \) nodes
Churn Definition and Examples

- Definition of churn

\[ C = \frac{1}{T} \sum_{\text{events}} \frac{|U_{i-1} \cap U_i|}{\max\{|U_{i-1}|, |U_i|\}} \]

- Where \( U_i \) = set of in use nodes after \( i \)th change
- \( U_0 \) = initial set
- \( \Delta = \text{Set Difference between } U_i \text{ and } U_{i-1} \)
- \( T = \text{time of some run} \)

- Suppose Run Length \( T \)
- Begin with \( k \) nodes
  - Two nodes fail
  - Two nodes added through reselection simultaneously

\[ C = \frac{1}{T} \left( \frac{2}{k} - \frac{2}{k} \right) \]

- Suppose \( k \) in-use nodes fail with no reselections

\[ C = \frac{1}{T} \left( \frac{1}{k} + \frac{1}{k-1} + \cdots + \frac{1}{1} \right) = \frac{1}{T} \ln k \]

Node Selection Strategies

- Node replacement strategies on two axis
  - Predictive / Agnostic
  - Replacement / Fixed

- Predictive Fixed
  - Observe nodes running
  - Use heuristic to decide a good fixed set

- Agnostic Fixed
  - Pick uniform-random nodes

- Predictive Replacement
  - Select a random initial set of nodes
  - Pick replacement after failure of in-use node

- Agnostic Replacement
  - Two different strategies
    - Upon failure pick a new random node (RR)
    - Given a ranking of nodes, replaced failed node with most preferred (PL)
    - Optimize for a metric other than churn

<table>
<thead>
<tr>
<th>Replacement</th>
<th>Fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predictive</td>
<td>Fixed</td>
</tr>
<tr>
<td>Predictive</td>
<td>Max Expectation</td>
</tr>
<tr>
<td>Predictive</td>
<td>Longest Uptime</td>
</tr>
<tr>
<td>Predictive</td>
<td>Optimal</td>
</tr>
<tr>
<td>Agnostic</td>
<td>Decent</td>
</tr>
<tr>
<td>Agnostic</td>
<td>Pick k random out of top 1/2</td>
</tr>
<tr>
<td>Agnostic</td>
<td>Most Available</td>
</tr>
<tr>
<td>Agnostic</td>
<td>Longest Lived</td>
</tr>
<tr>
<td>Agnostic</td>
<td>Random</td>
</tr>
<tr>
<td>Agnostic</td>
<td>Passive Preference List (PL)</td>
</tr>
<tr>
<td>Agnostic</td>
<td>Active Preference List (PL)</td>
</tr>
</tbody>
</table>
Trace Experiments

- **Synthetic Trace**
  - Pareto Distribution
    - Mean session time of 30 minutes
- **Planet Lab Ping**
  - Ping every 15 minutes over 18 month interval between pair of machines (200 – 400 nodes)
- **Web Site HTTP Request**
  - Request 129 websites every 10 minutes for 7 months from a single host
- **Microsoft Ping**
  - Ping over 51,000 desktops inside Microsoft every hour for 35 days
- **Skype superpeers**
  - Application level ping between 4000 nodes every 30 minutes for 25 days
- **Gnutella peers**
  - Send TCP connection request to over 17,000 IP address in the Gnutella P2P file sharing network every 7 minutes for 60 hours

<table>
<thead>
<tr>
<th>Trace</th>
<th>Length (days)</th>
<th>Mean # nodes up</th>
<th>Median node’s mean session time</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlanetLab</td>
<td>527</td>
<td>303</td>
<td>3.9 days</td>
</tr>
<tr>
<td>Web Sites</td>
<td>210</td>
<td>113</td>
<td>29 hours</td>
</tr>
<tr>
<td>Microsoft PCs</td>
<td>35</td>
<td>4175</td>
<td>8.8 days</td>
</tr>
<tr>
<td>Skype</td>
<td>25</td>
<td>710</td>
<td>11.5 hours</td>
</tr>
<tr>
<td>Gnutella</td>
<td>2.5</td>
<td>1646</td>
<td>1.8 hours</td>
</tr>
</tbody>
</table>

Additional Chord Experiment

- **Feed trace data through Chord Simulator**
- **Events**
  - Node Joins
  - Departures
  - Datagram Delivery
- **Once per second request to messages to owner of a single key**
- **Node failure unless both messages arrive at destination**
- **Split Trace in half**
  - Train on first half
  - Report statistics on second half
- **Demonstrate churn is proportional to fraction of failed requests in Chord**
**Experimental Results**

RR close to Max Exp when \( \alpha \) NOT small

Active and Passive PL similar to Fixed Random

---

**More Experimental Results**

RR is 1.2 – 3x better than Passive PL

RR is 2.5-10x better than Active PL

---

Figure 3: Churn of Random Replacement relative to other strategies. The key at right applies to all three plots.
Highlights of Node Selection Study

- Benefit of Replacement over Fixed Strategies
  » In two P2P traces, best fixed strategies match the best replacement strategies
  » Replacement offers 1.3 – 5x improvement over the best fixed strategies in other three traces

- Agnostic strategies
  » RR is worse for small $k$
    - usually within factor of 2 of Max Expectation
    - Max Expectation finds few nodes with very long time to fail

Analysis

- Expected Churn for all fixed results $= \frac{1}{k} \cdot \left( \frac{1}{k} \right) \cdot \text{number of failures}$
  » Nodes recover instantly

- Total Number of failures on $k$ nodes $= \frac{T_k}{u}$

- Expected Churn for fixed results $= \frac{2}{kT} \cdot \frac{T_k}{u} = \frac{2}{u}$

- Preference Lists
  » Passive PL similar to Fixed $= \frac{2}{u}$
  » Active PL requires additional $\frac{1}{kT}$
  » Active PL Expected Churn $= \frac{3}{u}$
More Analysis

- Intuition for Random Replacement
  » RR picks node v, after a failure, the replacements time to failure (TTF) is not drawn from the session time distribution \( f_i \)
  » RR is selecting the current session of a random node
  » Biased toward longer sessions
    - Node spends longer in a long session than a short one
  » Consider some node in a system
    - As it proceeds through the session, the probability that it has been picked by RR increases
    - Nodes with longer up times are more likely to be picked
    - For realistic distributions, nodes with longer uptimes are less likely to fail soon

- Unique solution for expected churn in RR
  \[
  E[C] = \frac{2}{\alpha d} \sum_{i=1}^{d} \frac{1}{\mu_i} \left( 1 - E \left[ \exp \left\{ \frac{-\alpha E[C] \cdot L}{2(1-\alpha)} \right\} \right] \right)
  \]

> Figure 5: Simulation and analysis of churn with varying session time distribution, \( \alpha = 0.1 \), and \( \alpha = 0.5 \).

- Corollary 1 States
  "The more skewed the distribution is the lower the expected churn of RR"
  - The greater the skew gives greater chance of selecting a longer session time.
  - Skewed distribution tend to be the case in realistic scenarios

- Corollary 2 States
  "If session times are stable and have equal mean, RR’s expected churn is no more than twice the expected churn of fixed or Preference"
  - Due to fact that equal mean session times simplifies equation in Theorem 1
    \[ E[C] \text{ of RR } \leq \frac{4}{\nu} \]
  - All Fixed or Preference List \( E[C] \geq \frac{2}{\nu} \)

From Appendix in Technical Report
Applications

- Connect to random anycast server
  - Replacement node strategy varies from RR to PL

- Select best next hop neighbor nodes in DHT Overlay
  - Chord simulation which modifies the finger table entries

- Join overlay multicast tree
  - Pick parent node based on several strategies

- Choose data replication policy using DHT Overlay
  - Study rate at which new replicas are created using Root or Random data replication policy

Anycast

- Application Scenario
  - End host connects to random server
  - When server fails
    - Obtain list of m server which have lowest latency
    - Connect to random one of these m
  - Continually probe for closer server $[0, t]$

- Experiment
  - Simulate number of failures in simple simulator at level of node joins and failures
  - Investigate various choices of m and t

  - Tradeoff between server failure rate and latency
    - Increase t moves from Active to Passive PL
    - Increase m moves to RR

  - Proposed Hybrid Strategy
    - $w \times$ latency $- (1-w) \times$ uptime
    - w decreases from 1 to 0
    - Moves from Passive PL to Longest Uptime

- Results
  - Increasing t from Active to Passive PL
    - 56% drop in failure rate
  - Increasing m from 1 to 32
    - Additional 13% drop in failure rate

(a) Stability-latency tradeoff in Skype trace
DHT Neighbor Topologies

Application Scenario
- Nodes in a DHT assigned an id(v) in keyspace
- Nodes maintain links to certain nodes based on id
  - Sequential Neighbors
  - Long Distance Neighbors
    - Finger table in Chord

Experiment
- Compare two different ways of selecting long distance neighbors
  - Active Preference List Strategy
  - Random Replacement Strategy

Chord Routing Example
- Instantiate a virtual ring of identifiers
  - Nodes join at specific identifiers
  - Based on hash of IP address and port
- Nodes maintain pointers to a set of nodes in the ring
  - Finger Table
  - Number of Finger Entries is log2 N
  - First entry is the successor node
- Each index is at least a power of 2 to the next successor node
- Node 2 queries for Key 26
  - 2 forwards to 22
  - 22 finds 26 in finger table
  - Sends location 26 back to 2
  - 2 sends next query to 26
  - 26 replies with location of request
- log2 N routing to destination
  - Cutting the distance in half with each step
- Active Preference List can be thought of as default Chord routing with k = 1
- Random Replacement strategy randomly selects which node to point to in a given finger table
  - Example: Choosing node 23 instead of 22 for 2+2
DHT Neighbor Topologies

Figure 7: DHT neighbor selection simulation in Gnutella trace.

Overlay Multicast Tree

- **Application Scenario**
  - Consider single-source multicast tree whose root does not fail.
  - When node $v$ joins, it contacts $m$ random suitable nodes.
  - $m$ is suitable if it is connected to the tree and has bandwidth for another child.
  - Pick one of those $m$ nodes based on a strategy:
    - Longest Uptime
    - Minimum Depth
    - Minimum Latency
  - When node fails:
    - Descendants experience interruption
    - Rejoin multicast stream

- **Experiment**
  - Report total number of interruptions
  - Sample mean node depth
    - Number of hops to root
  - Sample mean latency to root
  - Results
    - Case $m = 1$
      - Random Parent Selection
    - Increase $m$
      - Latency decreases to the root
  - Trees become less stable
    - Claim trees become less stable due to Preference List effect:
      - $m = n$ next node more likely to be neighbor
      - $m = 1$ more like RR
      - More stable
      - Grandchildren will experience fewer interruptions

- **Results**
  - Latency decreases to the root
  - Intermittent failures experience interruption
  - More stable
  - Grandchildren will experience fewer interruptions
Data Replication in DHTs

- Application Scenario:
  - Compare two different strategies for file replication in DHTs
    - Root Set
    - Random
  - Similar to Chord example, nodes are assigned IDs in a keyspace
  - Objects are assigned keys
    - key(o)
  - Node whose ID mostly closely follows key(o) is root(o)
  - Some k replicas are stored of o on some set of nodes
- Root Set Approach
  - Place replicas on k nodes the most closely follows key(o)
  - Passive PL
- Random Approach
  - Root r(o) stores directory of all available replicas of o
  - Contains two parameters k and 0 < f <= 1
  - Repair started when replicas fall below \[ f \cdot k \]

- Results:
  - Compare two strategies at \( f \in \{1, 0.75, 0.5\} \) and \( k \in \{2, \ldots, 20\} \)
  - Random requires 30% fewer replications than root set

Conclusions

- Provides overview of existing strategies for node selection
- Analytical results for Random Replacement
- Study of applications deploying spectrum of solutions based on current strategies for node selection
Comments

- Convinced that RR is useful?

- Fairness problems?

- How well will this strategy work once deployed?