The Physics of Peer-to-Peer Overlays

Supriya Krishnamurthy
KTH, School of Information and Communication Technology, Kista
and
Swedish Institute of Computer Science, Kista

Collaborators:

Sameh El-Ansary
SICS

Erik Aurell
KTH, Dept. of Computational Biology

Seif Haridi
SICS
KTH, ICT

1. The Statistical Theory of Chord Under Churn (IPTPS'05)
Overlay networks

Overlay Networks are networks working on top of the Internet and using the basic Internet Infrastructure
Motivation for having Overlays

- Need for improved services from the Internet (e.g. Better routing)
- Cost of modifying current Internet Infrastructure is prohibitive (also world wide coordinations among all ISPs required)
- For instance, servers on an Overlay network communicate with the help of routers but routing paths can be determined by the Overlay
- Make use of large glut of processing, memory and storage available
- Examples: Content Distribution Networks (Akamai), Improved Routing Networks (RON), File-Sharing networks (Gnutella, BitTorrent), P2P applications (where nodes are both clients as well as servers)

Figure taken from 'www.reviews.com/hottopic/hottopic_essay_01.cfm'
Your home computer could be

- Working for biotech, matching gene sequences
- Downloading telescope data
- Backing up others' files

Payments come in from biotech company, backup service etc

© Keith Ross and Dan Rubenstein
Slides marked with a ★ have figures taken from the tutorial by Ross and Rubenstein available online
Accessing and Storing Content
(how do we store and locate ever increasing amount of content?)

Content: specified by a (key, value) pair
Key: names of people  value: phone
Key: album name  value: address of available location/content

Option 1: Store them all in one computer -- the client server model

Appropriate Data Structure

Hash Table

(adding, deleting or searching: O(1))

Figure taken from:
en/wikipedia.org/wiki/Hash_table
A Good **Hash Function** is such that:
- The hash value is fully determined by the data being hashed
- The hash function uses all input data
- Data is uniformly distributed over entire set of possible hash values
- Similar strings have very different hash values

**Other Appropriate Data Structures**

- Binary Search Trees

```
50  75
20  | 66  80
  5  25 | 92  111
      166 200
```

- Skip Lists

```
3 7 12 17 19 21 25 26
```
What if the single server (in our example) wants to avoid getting besieged by requests from clients?

**Option 2:** Introduce a layer of caches between client and server.

- Server uses hash function to evenly distribute objects across caches
- Clients use the same function to discover which cache stores an object.

Partition the Hash table over different processors
Classical hash fns: \( x \rightarrow ax + b \mod p \), \( p \): number of processors

But what if \( p \) (the number of processors) is not fixed?

- Each time a new machine is added, nearly all keys get remapped
- A central server is required to keep track of \( p \) to inform the clients

Option 3: Consistent Hashing

1. Choose a hash fn. that maps keys to the range \([0,K]\)
2. Hash keys using \( h(x) \)
3. Hash processors using \( h(x) \)
4. Assign each key to nearest clockwise processor

Consistent Hashing and Random Trees: Distributed Caching Protocols for relieving hot spots on the World Wide Web, Karger et al
How should nodes search for a given item? How should they be connected?

**PRR Scheme**

- Map Nodes to \( b \)-ary numbers of \( m \) (e.g. \( b=2, m=4 \)) digits
- Node addressing defines nested groups
- Each node knows all the nodes in its inner group + Some delegate nodes in other groups
- Search proceeds by moving closer to target one digit at a time

Accessing Nearby Copies of Replicated Objects in a Distributed Environment: Plaxton, Rajaraman and Richa
Distributed Hash Tables (Structured P2P Overlays)

Key-Space partitioning

- DHTs use consistent hashing to partition a keyspace among a distributed set of nodes.
- A metric topology is defined on the space of nodes and keys.
- Each node is responsible for storing the keys “closest” to it.

Chord, Pastry/Tapestry, Koorde
Kademlia, Symphony, Viceroy (Circle) CAN (d-dimensional torus)
Distributed Hash Tables (Structured P2P Overlays)
Connections between nodes

In all these systems, nodes are connected to a set of immediate neighbours. 

In addition:

**chord:**

![chord diagram]

**viceroy:**

![viceroy diagram]

**Finger Table Entries for Node 1**

<table>
<thead>
<tr>
<th>Start</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>+2⁰</td>
<td>31</td>
</tr>
<tr>
<td>+2¹</td>
<td>31</td>
</tr>
<tr>
<td>+2²</td>
<td>31</td>
</tr>
<tr>
<td>+2³</td>
<td>31</td>
</tr>
<tr>
<td>+2⁴</td>
<td>31</td>
</tr>
<tr>
<td>+2⁵</td>
<td>50</td>
</tr>
<tr>
<td>+2⁶</td>
<td>68</td>
</tr>
</tbody>
</table>
Distributed Hash Tables (Structured P2P Overlays)

Routing Algorithms

**Chord**

**Greedy Routing**

**Viceroy**

**Butterfly Routing**

**CAN**

Greedy Routing on Cartesian space

**Pastry**

**Tapestry**

**PRR routing**
Unstructured P2P: Popular File Sharing

**NAPSTER**

**GNUTELLA**

**KAZAA**

**BitTorrent**

1. GET file.torrent
2. GET
3. list of peers
4. file.torrent info:
   - length
   - name
   - hash
   - url of tracker
Earlier approaches have either been empirical or have focussed on proving theoretical bounds.

By looking at the full time-evolution of the system, we can predict performance of Chord to better than 1% accuracy for any value of the system parameters.

The analysis can take into account fine details as well as broad features.
Parameters defining the System

Periodic Maintenance
• Rate of periodically checking a successor or a finger

Reactive Maintenance
• Rate of periodically checking successor
• Rate of sending messages once error is found

K = 128
N = 12
(S, Fin1, Fin2, Fin3, ... Fin7) = (A, A, A, A, A, D, A)
Overlay Networks: Analysis

We are primarily interested in evaluating the performance of the network, e.g. How many hops does a Lookup take on average?

Greedy Search on a ring-based Overlay
Since the length of the Lookups depends on the state of the pointers, we need to estimate the probability that each of the successors and fingers are correct, incorrect or dead.

If we take any one pointer in the system, its evolution in time looks something like this:

![Diagram showing pointer states with transitions: join, stabilisation, fail]

We would like to estimate the fraction of red nodes in the system,

The probabilities are evaluated as a function of the churn.

We then derive an equation for the average Lookup length, which uses the above probabilities as input.
**Probability of Failed or Outdated First Successor Pointers**

- **W₁**: the number of wrong (failed or outdated) s₁ pointers
- **D₁**: number of failed s₁ pointers

\[
\frac{dW_1}{dt} = \lambda_j N (1 - w_1) + \lambda_f N (1 - w_1)^2 - \lambda_f N w_1^2 - \alpha \lambda_s N w_1
\]

\[
w_1(r, \alpha) = \frac{2}{3 + r\alpha} \approx \frac{2}{r\alpha}; \quad d_1(r, \alpha) \sim \frac{1}{2} w_1(r, \alpha)
\]
Theory and Simulation for $w_1(r, \alpha)$, $d_1(r, \alpha)$, $I(r, \alpha)$

- $w_1(r, \alpha)$ ≡ fraction of incorrect first successor pointers
- $d_1(r, \alpha)$ ≡ fraction of failed first successor pointers
- $I(r, \alpha)$ ≡ fraction of lookups which give inconsistent answers

\[
I(r, \alpha) = w_1(r, \alpha) - d_1(r, \alpha) \sim \frac{1}{\alpha r}
\]
Number of failed $k^{th}$ Fingers: $F_k$

Definition: For a node $n$, the $k^{th}$ finger is the first node succeeding $n + 2^{k-1}$, $1 \leq k \leq \mathcal{M}$.

\[
\begin{align*}
F_k(t + \Delta t) &= F_k(t) + 1 \\
&= F_k(t) - 1 \\
&= F_k(t) + 1 \\
&= F_k(t) + 3 \\
&= F_k(t)
\end{align*}
\]

Prob. of Change

\[
(\lambda_j N \Delta t) \sum_{i=1}^{k} p_{join}(i, k) f_i
\]

\[
(1 - \alpha) \frac{1}{\mathcal{M}} f_k (\lambda_s N \Delta t)
\]

\[
(1 - f_k)^2 [1 - p_1(k)] (\lambda_f N \Delta t)
\]

\[
(1 - f_k)^2 (p_1(k) - p_2(k)) (\lambda_f N \Delta t)
\]

\[
(1 - f_k)^2 (p_2(k) - p_3(k)) (\lambda_f N \Delta t)
\]

otherwise

To $O(1/r)$, $f_k \approx \frac{(1 + \tilde{p}_{rep}(k)) \mathcal{M}}{(1 - \alpha)r}$. 
Theory and Simulation for $f_k(r, \alpha)$: The fraction of failed $k$th fingers
Lookup Hop Count: Theory

The Cost (Latency) for a node $n$ to lookup a target $t = \xi + m$ is:

$$
C_{\xi+m} = C_\xi [1 - a(m)] + (1 - f_k)a(m) \left[ 1 + \sum_{i=0}^{m-1} bc(i, m)C_{m-i} \right] + f_k a(m) \left[ 1 + \sum_{i=1}^{k-1} h_k(i) \sum_{l=0}^{\xi/2^i-1} bc(l, \xi/2^i)(1 + (i - 1) + C_{\xi-l+m}) + O(h_k(k)) \right]
$$

where $\xi = 2^{k-1} + n$, $\xi_i \equiv \sum_{m=1,i} \xi/2^m$ and

$$
h_k(i) = a(\xi/2^i)(1 - f_{k-i}) \Pi_{s=1,i-1}(1 - a(\xi/2^s) + a(\xi/2^s)f_{k-s}) \quad i < k
$$

$$
h_k(k) = \Pi_{s=1,k-1}(1 - a(\xi/2^s) + a(\xi/2^s)f_{k-s})
$$
Results

Comparison of theory and Simulations for N=1000

Prediction of Performance for higher N based on functional form
$L \sim \frac{1}{2} \log_2 N (1 + f(r) + 3f(r)^2)$
We can predict Performance for different routing table sizes (varying number of connections per node)

We can also predict Performance for different maintenance strategies (active vs. reactive)
Conclusions

- Several applications have been proposed for P2P systems: Data storage systems, Web publishing and caching, network performance measurements, managing flash crowds, group communication.

- But making these systems “secure” is a challenge.

- In addition, “free loading” could be a problem: users need incentives to cooperate.

- Though these systems are engineered to work in a dynamic environment, what if users come and go every 3 minutes (from Kazaa measurements)?

- In one project at MIT, even extremely technical savvy users did not trust that their data, no matter how strongly encrypted, could be stored securely on other people's untrusted machines!
Future Directions

Lookup *Latency* is a better measure of performance.

To take the *time* into account, we need to generalise the theory so that lookups are not *instantaneous*.

If we do this, we can also take *proximity* into account.

- What if a certain fraction of the links are slow and the rest are fast?
- What if proximity is based not just on geographical distances but other metrics as well?
- How will the maintenance protocol used affect results?
- Are there other maintenance protocols of interest, besides the active and reactive schemes already analysed?