Kinetic theory of (simple)
Distributed Information Systems

Erik Aurell
Dept Information and Computer Science, TKK, Espoo, Finland
Dept Computational Biology, KTH, Stockholm, Sweden


Motivations
Heterogeneity, real-time sustainable performance protection, scalability, data floods and operating expenditures of modern telecommunication systems are driving research into new architectures for future Autonomic Network Management Systems. […] Without global real-time coordination, they must continue to operate correctly in the presence of inconsistency, incomplete FCAPS data and badly formed Network Management Policies […]
Overlays
Overlay networks

Overlay Networks are networks working on top of the Internet and using the basic Internet Infrastructure.
Pros and Cons of a Client Server Architecture vs an Overlay

**Advantages**

- Content is reliably available
- Local failures have local effects
- Hosts can be optimised for their specific roles (high-performance server machines with high bandwidth connectivity)
- Resource location is hierarchical; different degrees of trust with different roles

**Disadvantages**

- Scalability (ability of network to support increasing use) is hard to achieve
- Servers present single point of failure: Not fault tolerant
- Requires administration
- Unused resources at the network edge

("The power of the network increases exponentially by the number of computers connected to it. Therefore, every computer added to the network both uses it as a resource while adding resources, in a spiral of increasing value and choice --- Robert M. Metcalfe")
Unstructured P2P: Popular File Sharing

**NAPSTER**

- Query and results

**KAZAA**

- Supernodes

**GNUTELLA**

**BitTorrent**

1. GET file.torrent
2. GET
3. List of peers
4. URL

file.torrent info:
- length
- name
- hash
- URL of tracker
Storing, finding, hashing, structured overlays & Distributed Hash Tables
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

Figure taken from: en.wikipedia.org/wiki/Hash_table
What if the single server (in our example) wants to avoid getting too many 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

```
<table>
<thead>
<tr>
<th>Key</th>
<th>Hash Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
Machine Id
```

![Diagram showing hash function and machine id]

Classical hash fns: $x \mapsto ax + b \pmod{p}$, $p$: 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
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
Third generation of P2P systems (structured overlays)

Main representatives:
Chord, Pastry, Tapestry, CAN, Kademlia, P-Grid, Viceroy
Distributed Hash Tables (Structured P2P Overlays)

Routing Algorithms

Chord

Greedy Routing

125
105
90
84
79
68
64
57
50
23
1

CAN

Greedy Routing on Cartesian space

Viceroy

Butterfly Routing

level 1
level 2
level 3
level 4

Pastry

Tapestry

PRR routing

Route(d46a1c)

65a1fc

d13da3

d4213f

d467c4

d462ba

d46

d47

www.cs.toronto.edu/~stefan/courses/csc2231/05a
Chord under churn
Problem: While all bad edges are eventually hopefully eliminated, they affect the performance of the system. The more often a lookup stumbles upon a bad edge, the worse the performance of the system. So which rate of stabilization against which rate of churn, would lead to how good a performance?
Key look-up in Chord (circle topology)

Key space $K$ nodes

Routing table $M = \log K$

Actually, one RT to successors [not shown], and one RT pointing to successors of nodes halfway around the ring, etc.
Key look-up in Chord (circle topology)

Key space $K$ nodes

Routing table $M = \log K$

Actually, one RT to successors [not shown], and one RT pointing to successors of nodes halfway around the ring, etc.
Key look-up in Chord (circle topology)

Key space $K$ nodes

Routing table $M = \log K$

Actually, one RT to successors [not shown], and one RT pointing to successors of nodes halfway around the ring, etc.
Key look-up in Chord (circle topology)

Item found in order log $K$ hops (3 in this example)

In a “true” system, number of participating peers $N$ is much less than $K$, and behavior depends more on $N$ than on $K$

Assuming a uniform distribution key look-up should take on average $\frac{1}{2}\log N$ hops
Maintenance Strategies

Periodic Maintenance

Reactive Maintenance
Deployed P2P systems operate under substantial churn.

A generic curve obtained in several empirical studies of DHTs, e.g., Li et al (IPTPS’04), Rhea et al (USENIX 2004), and by us.
Kinetic theory (simple)
Kinetic theory of Chord under churn

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

A detailed description keeps track of the states (correct, wrong, dead) of all pointers in the system.

In analogy to the well-stirred ansatz in chemical kinetics, count only the numbers of pointers of all states and kinds.

The right-hand side of the master equation is partly treated exactly, partly a factorized approximation is used.
Parameters defining the System

**Periodic Maintenance**

- Rate of periodically checking
- a successor or a finger

\[ \lambda_J : \text{Rate of Joins} \]
\[ \lambda_F : \text{Rate of Failures} \]
\[ \lambda_S \alpha : \text{Rate of checking Successors} \]
\[ \lambda_S (1-\alpha) : \text{Rate of checking Fingers} \]

**Reactive Maintenance**

- Rate of periodically checking successor
- Rate of sending messages once error is found

\[ \lambda_J : \text{Rate of Joins} \]
\[ \lambda_F : \text{Rate of Failures} \]
\[ \lambda_S : \text{Rate of checking Successors} \]
\[ \lambda_M : \text{Rate of sending messages} \]

\( K=128 \)
\( N=12 \)
\( (S, Fin1, Fin2, Fin3, ..., Fin7) = (A, A, A, A, A, A, D, A) \)
**Probability of Failed or Outdated First Successor Pointers**

<table>
<thead>
<tr>
<th>Change in $W_1(r, \alpha)$</th>
<th>Prob. of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W_1(t + \Delta t) = W_1(t) + 1$</td>
<td>$(\lambda_j N \Delta t)(1 - w_1)$</td>
</tr>
<tr>
<td>$W_1(t + \Delta t) = W_1(t) + 1$</td>
<td>$\lambda_f N(1 - w_1)^2 \Delta t$</td>
</tr>
<tr>
<td>$W_1(t + \Delta t) = W_1(t) - 1$</td>
<td>$\lambda_f N w_1^2 \Delta t$</td>
</tr>
<tr>
<td>$W_1(t + \Delta t) = W_1(t) - 1$</td>
<td>$\alpha \lambda_s N w_1 \Delta t$</td>
</tr>
<tr>
<td>$W_1(t + \Delta t) = W_1(t)$</td>
<td><em>otherwise</em></td>
</tr>
</tbody>
</table>

$W_1 \equiv$ the number of wrong (failed or outdated) $s_1$ pointers

$D_1 \equiv$ number of failed $s_1$ 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 $w_1(r, \alpha)$, $d_1(r, \alpha)$, $I(r, \alpha)$

- $w_1(r, \alpha) \equiv$ fraction of incorrect first successor pointers
- $d_1(r, \alpha) \equiv$ fraction of failed first successor pointers
- $I(r, \alpha) \equiv$ 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: $F_k$**

*(Periodic Maintenance Strategy)*

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

<table>
<thead>
<tr>
<th>$F_k(t + \Delta t)$</th>
<th>Prob. of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>$= F_k(t) + 1$</td>
<td>$(\lambda_j N \Delta t) \sum_{i=1}^{k} p_{\text{join}}(i, k) f_i$</td>
</tr>
<tr>
<td>$= F_k(t) - 1$</td>
<td>$(1 - \alpha) \frac{1}{M} f_k(\lambda_s N \Delta t)$</td>
</tr>
<tr>
<td>$= F_k(t) + 1$</td>
<td>$(1 - f_k)^2[1 - p_{1}(k)](\lambda_f N \Delta t)$</td>
</tr>
<tr>
<td>$= F_k(t) + 2$</td>
<td>$(1 - f_k)^2(p_{1}(k) - p_{2}(k))(\lambda_f N \Delta t)$</td>
</tr>
<tr>
<td>$= F_k(t) + 3$</td>
<td>$(1 - f_k)^2(p_{2}(k) - p_{3}(k))(\lambda_f N \Delta t)$</td>
</tr>
<tr>
<td>$= F_k(t)$</td>
<td><em>otherwise</em></td>
</tr>
</tbody>
</table>
**Number of failed \( k^{th} \) Fingers: \( F_k \)**

(Reactive Maintenance Strategy)

<table>
<thead>
<tr>
<th>( F_k(t + \Delta t) )</th>
<th>Prob. of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>( = F_k(t) + 1 )</td>
<td>( c_{3.1} = (\lambda_j N \Delta t) \sum_{i=1}^{k} p_{join}(i, k) f_i )</td>
</tr>
<tr>
<td>( = F_k(t) - 1 )</td>
<td>( c_{3.2} = \frac{f_k}{\sum_k f_k} (\lambda_M N S_2 (1 - w'_1) A(w_1, w'_1) \Delta t) )</td>
</tr>
<tr>
<td>( = F_k(t) + 1 )</td>
<td>( c_{3.3} = (1 - f_k)^2 [1 - p_1(k)] (\lambda_f N \Delta t) )</td>
</tr>
<tr>
<td>( = F_k(t) + 2 )</td>
<td>( c_{3.4} = (1 - f_k)^2 (p_1(k) - p_2(k)) (\lambda_f N \Delta t) )</td>
</tr>
<tr>
<td>( = F_k(t) + 3 )</td>
<td>( c_{3.5} = (1 - f_k)^2 (p_2(k) - p_3(k)) (\lambda_f N \Delta t) )</td>
</tr>
<tr>
<td>( = F_k(t) )</td>
<td>( 1 - (c_{3.1} + c_{3.2} + c_{3.3} + c_{3.4} + c_{3.5}) )</td>
</tr>
</tbody>
</table>
Theory and Simulation for $f_k(r, \alpha)$: The fraction of failed $k$th fingers

![Graph showing the comparison between simulation and theory for $f_k(r, \alpha)$]
Lookup Hop Count: Theory

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

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

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

$$
h_k(i) = a(\xi/2^i)(1 - f_{k-i})\prod_{s=1,i-1}^{1}(1 - a(\xi/2^s) + a(\xi/2^s)f_{k-s}) \quad i < k \\
h_k(k) = \prod_{s=1,k-1}^{1}(1 - a(\xi/2^s) + a(\xi/2^s)f_{k-s})
$$
Lookup latency, simulations and theory

Compared well with the generic curve obtained in several empirical studies of DHTs, e.g., Li et al (IPTPS’04), Rhea et al (USENIX 2004), and by us prior to the theory.
DKS is a system developed at SICS, which is similar to Chord, but is based on K-ary search.

Large $b$ is preferred at low churn, while $b=2$ is competitive at high churn. See also Li et al IPTPS’02.
Correction-on-Change, used in DKS is an example of reactive maintenance. C-on-C outperforms periodic maintenance at low churn, but hits a barrier at high Churn, where it cannot keep up (El-Ansary, 2003).
Future directions
Why does the analysis work as well as it does?

The well-stirred assumption takes all nodes equivalent. Under independent joins and failures, this is true, but it would be wrong if failures were correlated. This may be an issue for proximity-aware congestion-sensitive systems.

These errors could perhaps be estimated....
Why does the analysis work as well as it does?

Every failure or join changes all pointers which point to, or should point to, that node, i.e. many correlated changes. The performance measures considered however do not depend on more than one member from each such set.

This should depend on some overlap....
Why does the analysis work as well as it does?

We only compute the average number of pointers in each state, not the variance, nor the full distribution. The likelihood of ring break-up in a Chord-like system, with $\log(N)$ successors in the routing table, was estimated to be of order $(1/r)^{\log(N)}$, where $r$ is the rate of churn in the system. An analogous relation has been numerically tested for a system with a small routing table of size 2.

Ring break-up is rare event, hence a Large Deviation…
Including latency, gives a phase transition


Lookup *Latency* is another measure of performance, and also strongly interferes with the self-healing of the overlay.

Krishnamurthy and Ardelius have generalised the theory so that lookups are not *instantaneous*.

By doing this, they were also also to take *proximity* into account, and analysed two algorithms used by proximity-aware networks.

The model analysed by Krishnamurthy and Ardelius has a simple bimodal distribution of delays, where a fraction $p$ of the links are slow, and $1-p$ are fast.
At larger system sizes there is a parameter range where maintenance of long fingers does not keep up with network failures.

The small-world property (log N hop count) is then lost.

Using master equations to analyse aggregation of network data, for distributed FCAPS

What is the "best" way of aggregating network data in a large scale + dynamic network?

- Spanning trees? Peer-to-peer interactions?
- Performance, robustness, scope

What is the effect of node dynamicity = churn?

- Models and performance analysis needed

P2P content distribution: should distributed overlay construction and distributed data forwarding interact in some way?

Distributed overlay construction: nodes select neighbors based on some local decision. Neighbor relations give the overlay graph.

Distributed data forwarding: data upload/download (push/pull) is decided locally at the overlay nodes, limited by the overlay graph.

Thus the way the overlay is constructed affects the data distribution efficiency. Do we need cross-layer optimization or can we deal with the two issues separately?

E. Aurell, V. Fodor, P. Holme, G. Karlsson and S. Krishnamurthy [in preparation].
thanks to

Sameh El-Ansary
Supriya Krishnamurthy
Seif Haridi

John Ardelius

Gunnar Karlsson
Rolf Stadler
Mads Dam
Viktoria Fodor