Evaluating Performance in (Structured Peer-to- Peer) Overlay Networks

Supriya Krishnamurthy

Collaborators:

Sameh El- Ansary

Erik Aurell

Seif Haridi

John Ardelius

SWEDISH INSTITUTE OF COMPUTER SCIENCE SICS

KTH VETENSKAP OCH KONST

KTH Informations- och kommunikationsteknik

1. *The Statistical Theory of Chord Under Churn* (IPTPS'05)
2. *An Analytical Study of a Structured Overlay in the Presence of Dynamic Membership* (IEEE/ACM -TON,2008)
3. *Comparing Maintenance Strategies for Overlays* (Proceedings of PDP2008, Toulouse)
4. An Analytical Framework for evaluating Performance in Proximity-aware Overlay Networks (Draft)

Networks

- A network is a set of vertices (nodes) and connections (edges) between them.
- Examples include the Internet, Social Networks, neural networks, metabolic networks, food webs, citation networks, *etc*
- Current research focuses on :

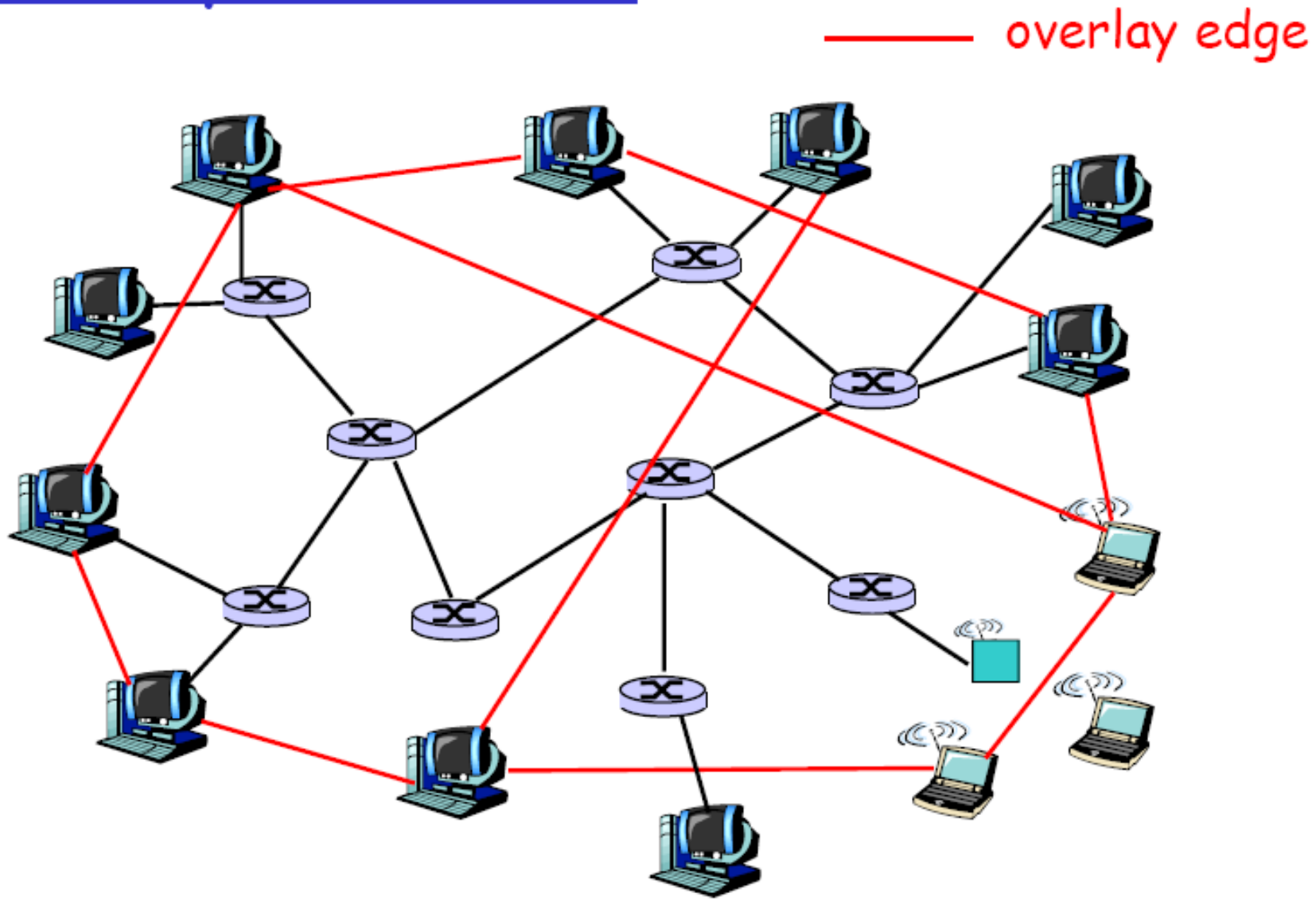
1) Finding quantities which describe network properties and suggesting ways to measure them.

2) Creating models of evolving networks that help understand what sort of evolution rules (or optimization strategies) give the above properties.

3) Predicting behaviour of networked systems based on structural properties and local rules (How will network structure affect some process occurring on the network?)

- This talk related to 3) with added complication that the process is occurring on an evolving network.

Overlay networks



- An overlay network is a virtual network of nodes and logical links that is built on top of an existing network with the purpose to implement a network service that is not available in the existing network (I. Stoica)

Peer-to-Peer

- A distributed network architecture may be called a Peer-to-Peer network, if the participants share a part of their own hardware resources (processing power, storage capacity, network link capacity).

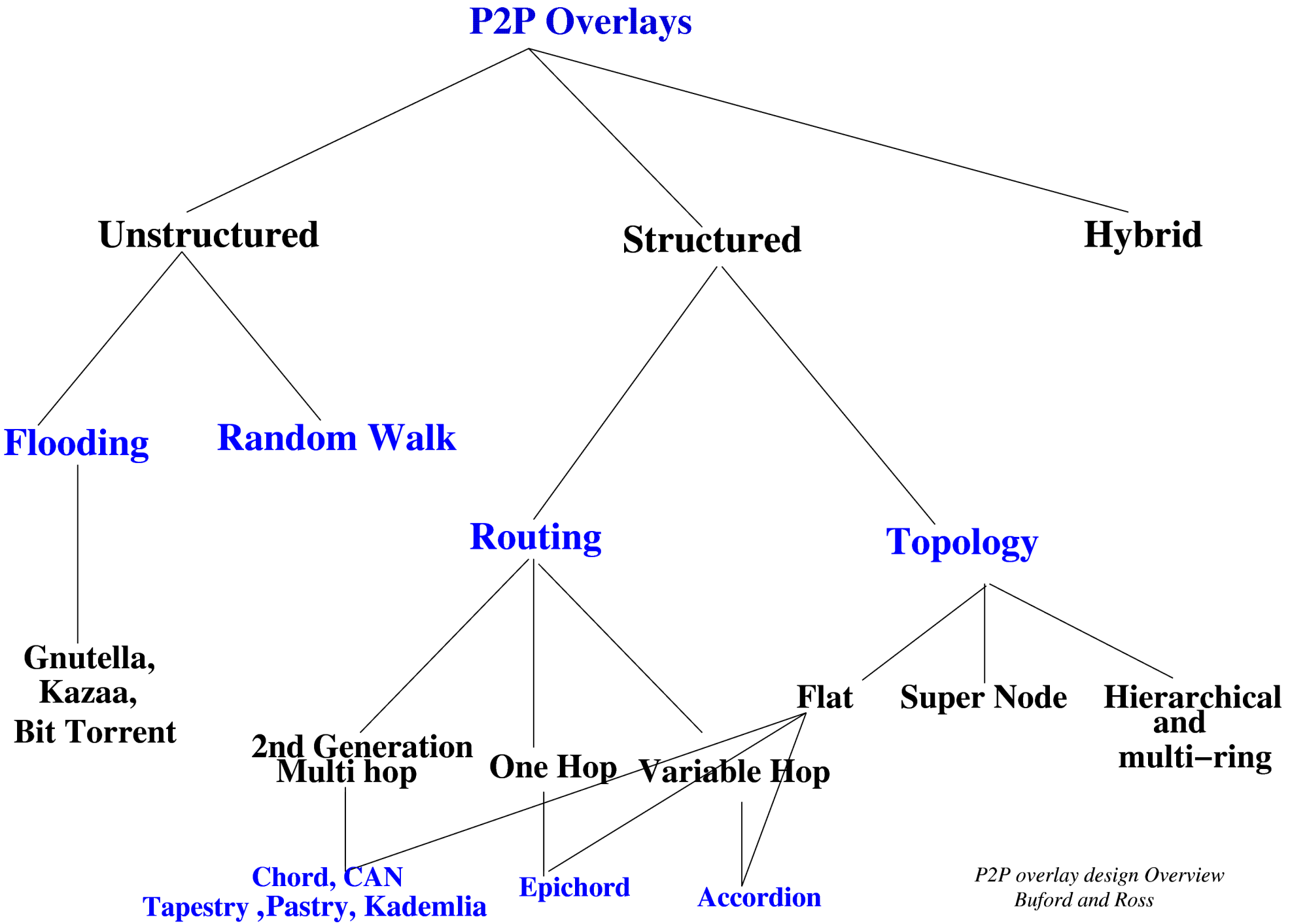
- Network is highly dynamic: peers are highly transient

- Fault-tolerant architecture: No single point of failure

- Potential to enable rapid and low-cost deployment of large-scale applications

But.....

- P2P systems are at the mercy of their user population in terms of both system performance and the extent and variety of available content

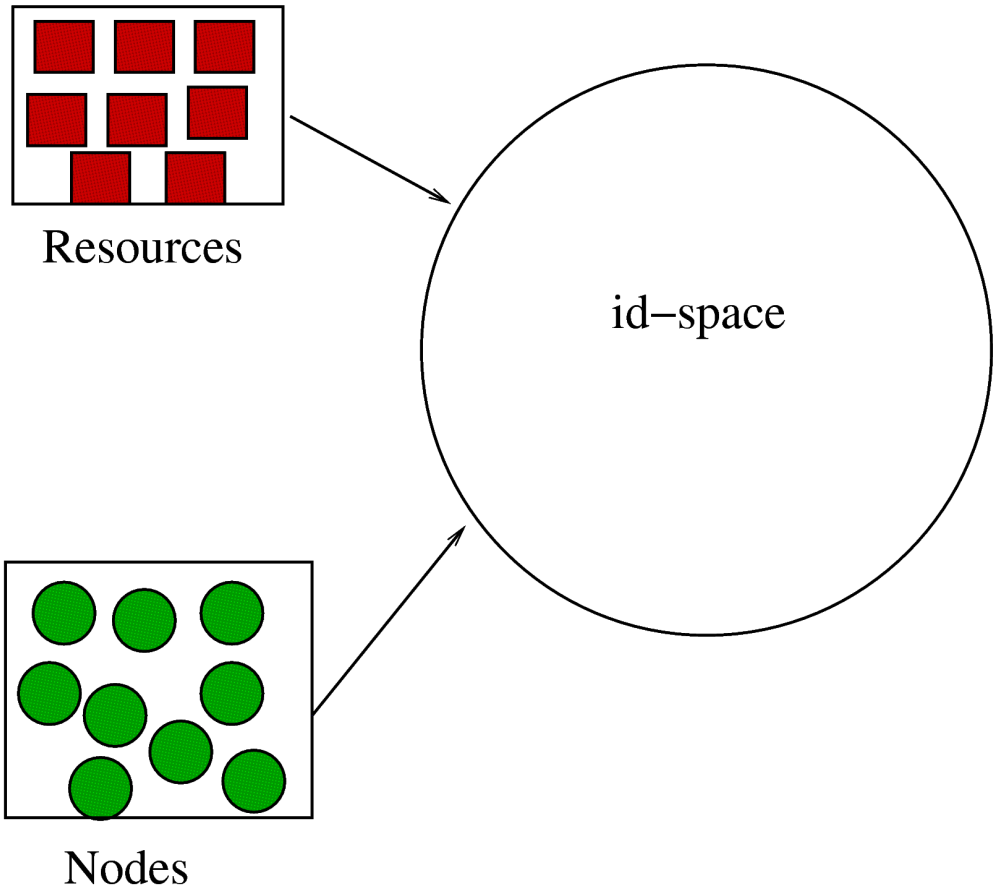


P2P overlay design Overview
Buford and Ross

Distributed Hash Tables(Structured P2P Overlays)

- Choice of Identifier space → **[0, K-1]**

- Mapping of resources and peers to the identifier space

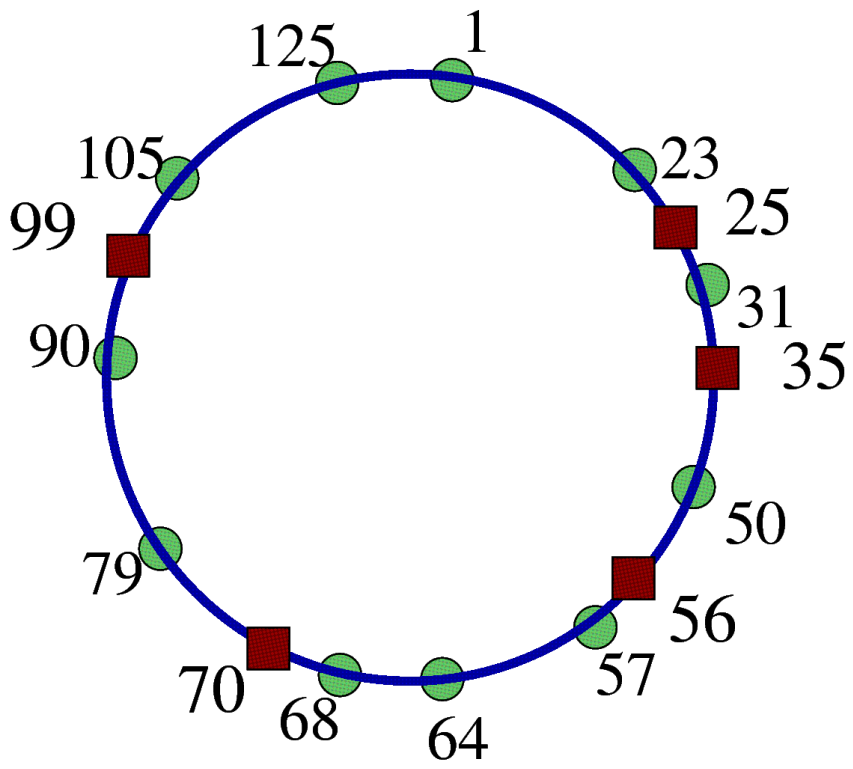


Distributed Hash Tables(Structured P2P Overlays)

• Choice of Identifier space → **[0, K-1]**

• Mapping of resources and peers to the identifier space

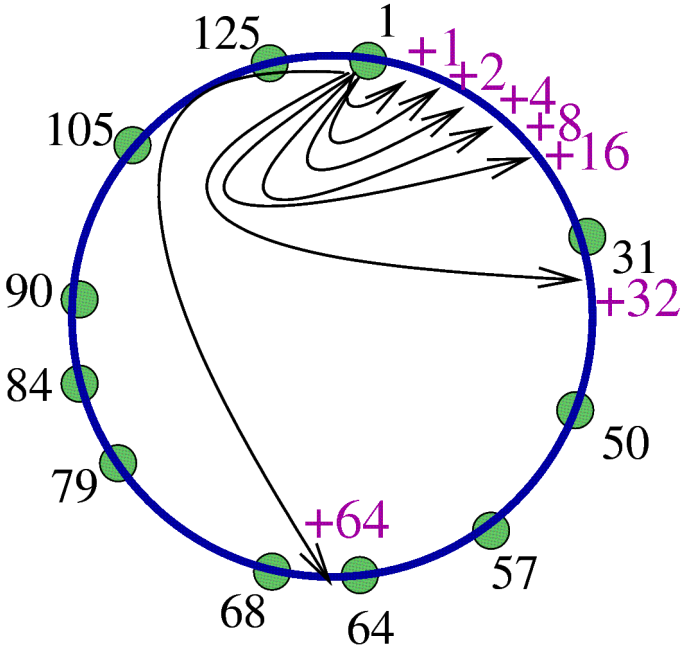
•Management of identifier space by nodes (so that when nodes join or fail, the effect is 'local')



Distributed Hash Tables(Structured P2P Overlays)

• Connect Nodes+ Specify Routing strategy
(design a network which allows fast searches using a simple greedy algorithm)

• Eg. Chord



Finger Table Entries for Node 1

Start	Node
$+2^0$	31
$+2^1$	31
$+2^2$	31
$+2^3$	31
$+2^4$	31
$+2^5$	50
$+2^6$	68

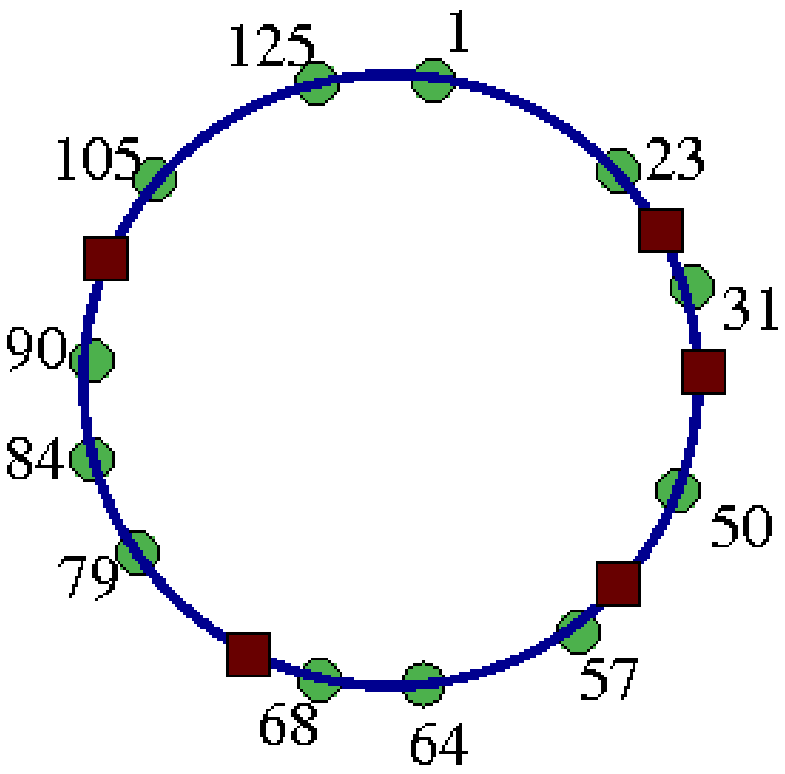
•Routing strategy: Go as close to target as possible, without overshooting

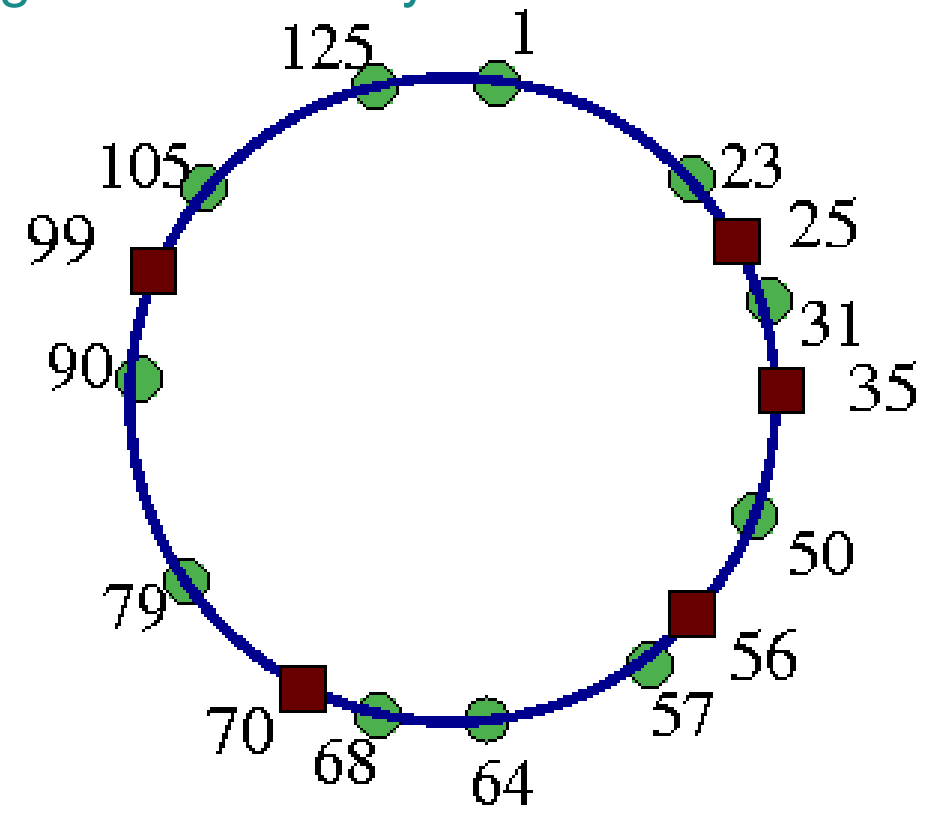
• Maintenance Strategy: Periodically check all connections
(necessary in the face of node joins and failures: *Churn*)

Performance Evaluation

- Churn affects the number of 'wrong' connections in the network
- Wrong connections delay lookups which causes poor performance

Greedy Search on a ring-based Overlay

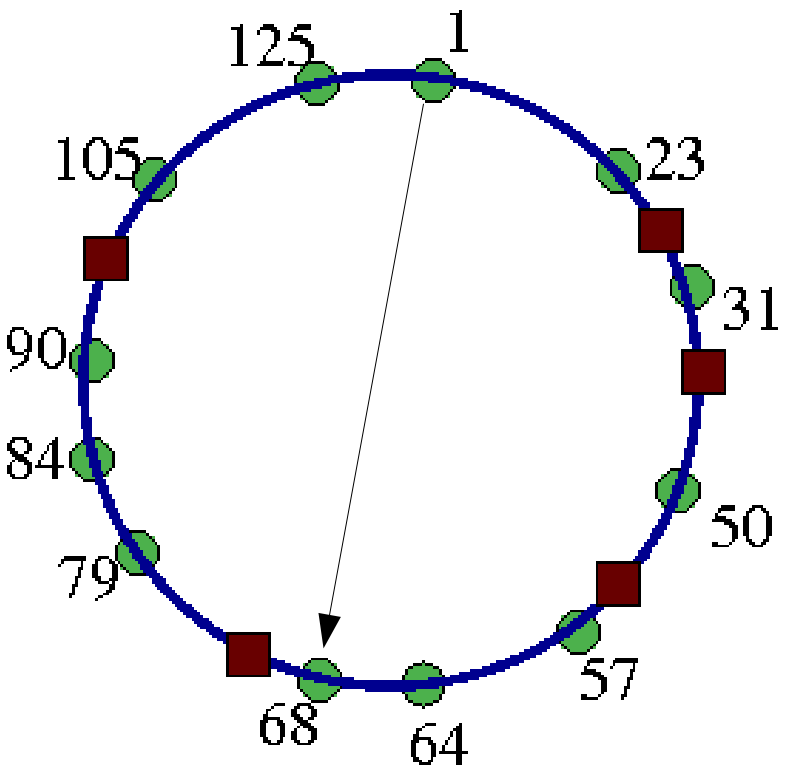


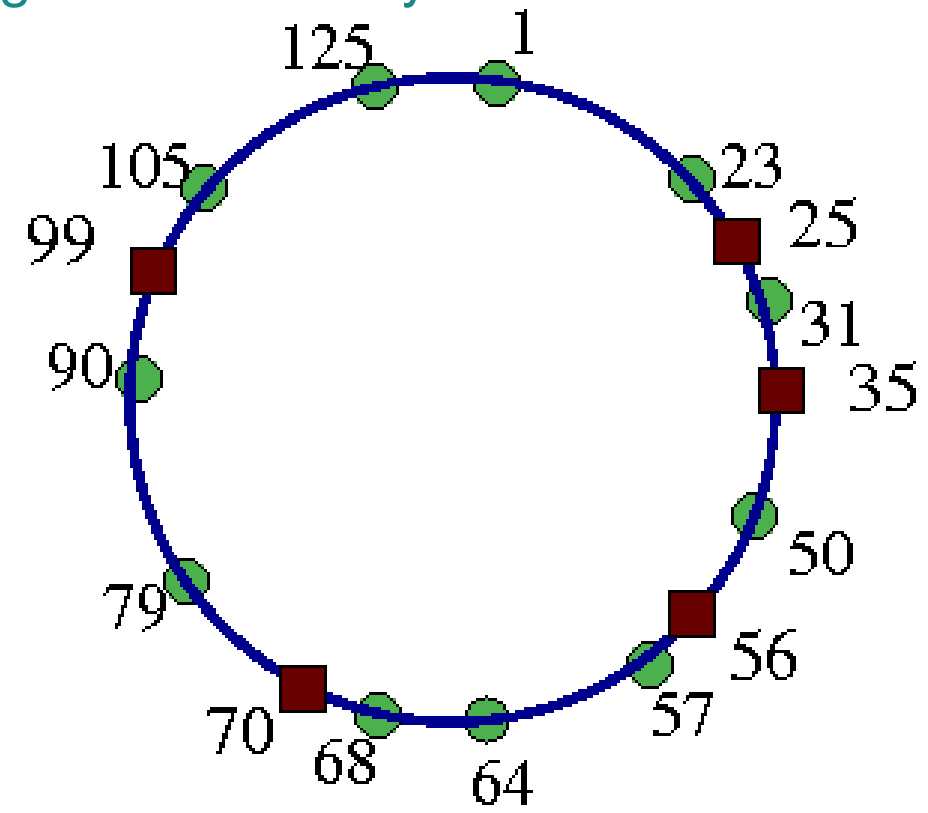


Performance Evaluation

- Churn affects the number of 'wrong' connections in the network
- Wrong connections delay lookups which causes poor performance

Greedy Search on a ring-based Overlay

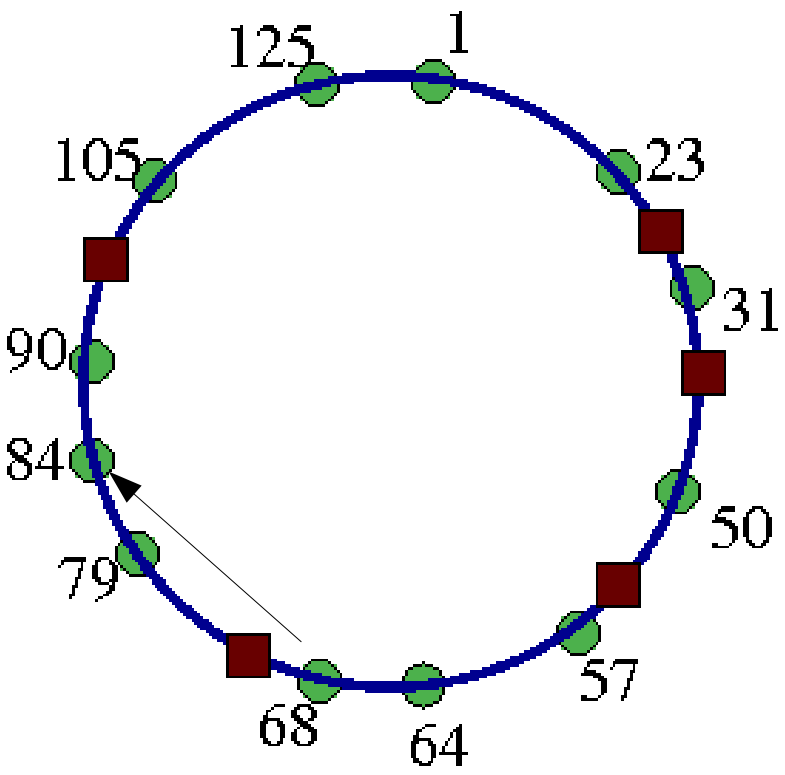


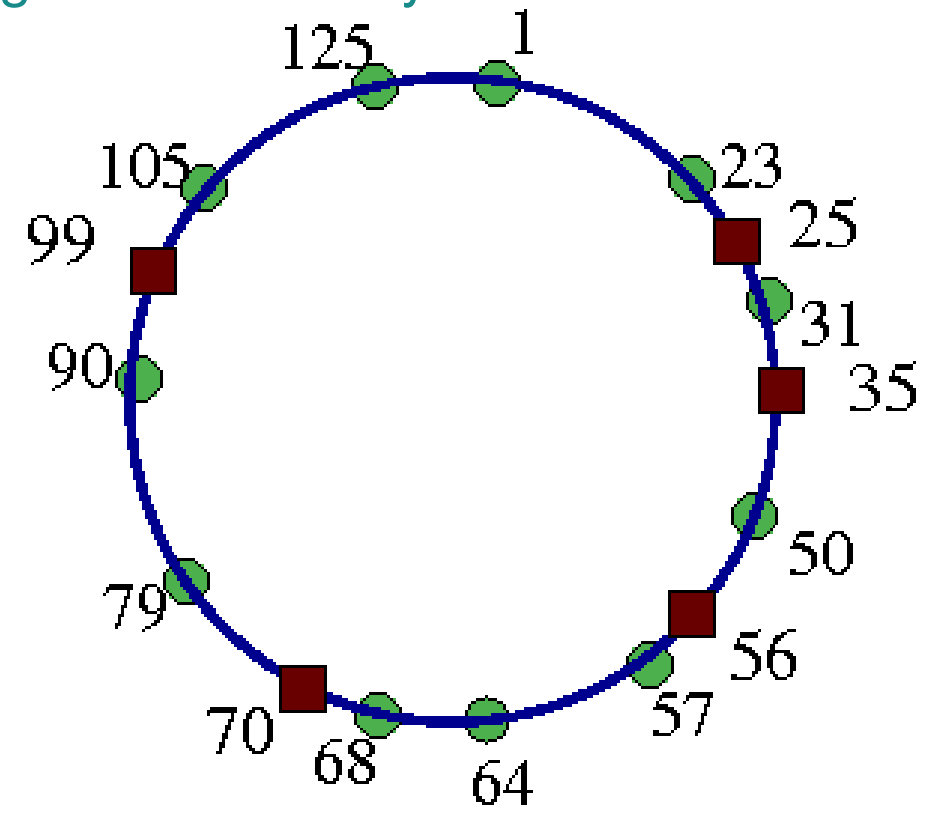


Performance Evaluation

- Churn affects the number of 'wrong' connections in the network
- Wrong connections delay lookups which causes poor performance

Greedy Search on a ring-based Overlay

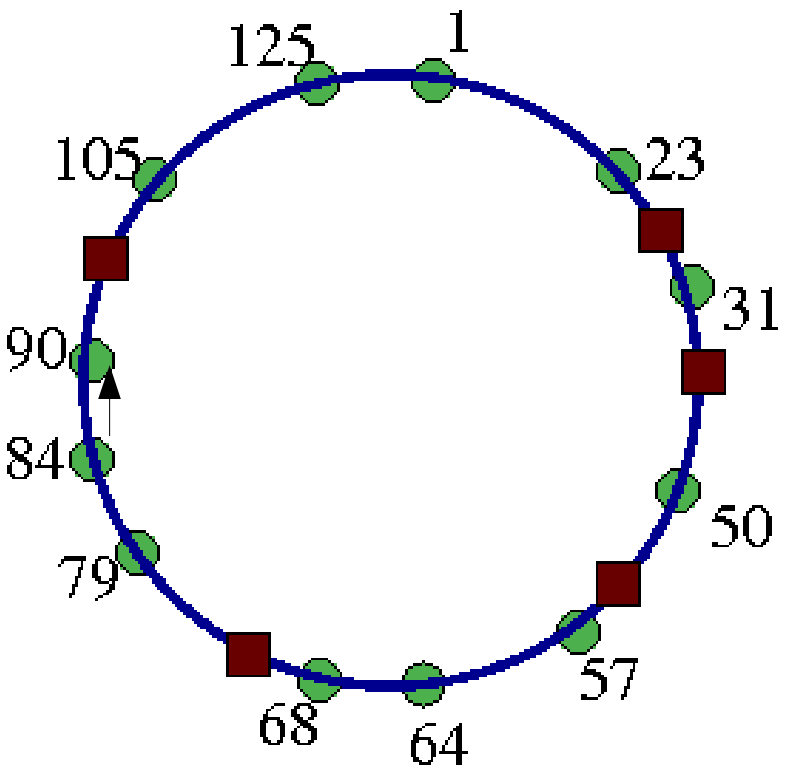


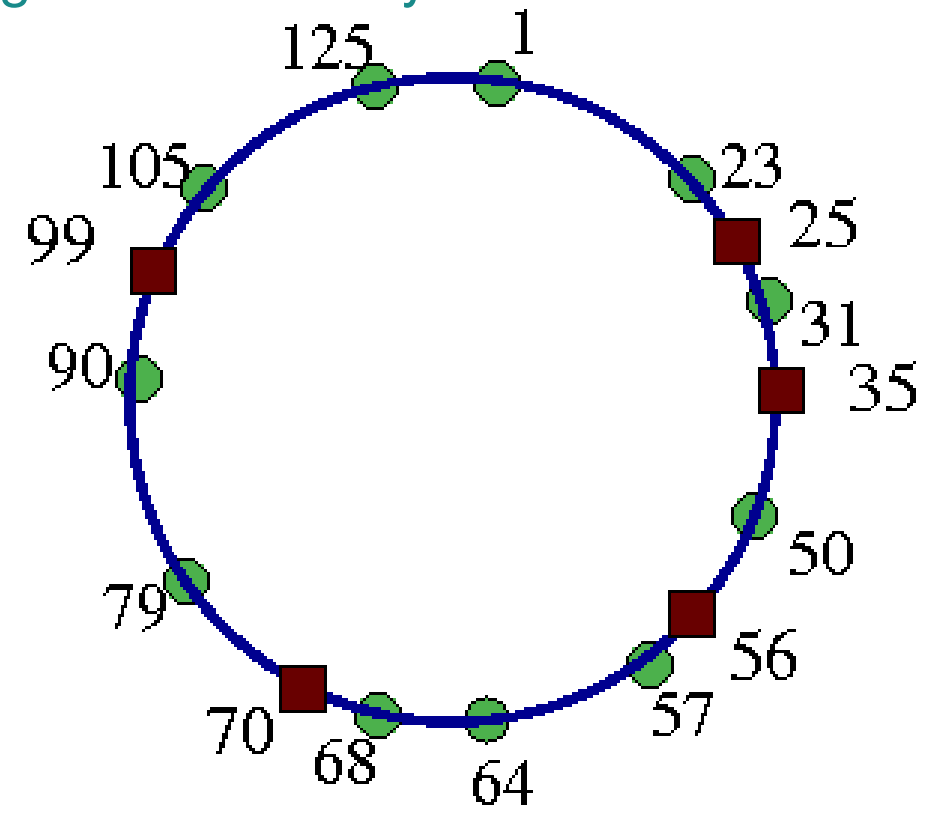


Performance Evaluation

- Churn affects the number of 'wrong' connections in the network
- Wrong connections delay lookups which causes poor performance

Greedy Search on a ring-based Overlay

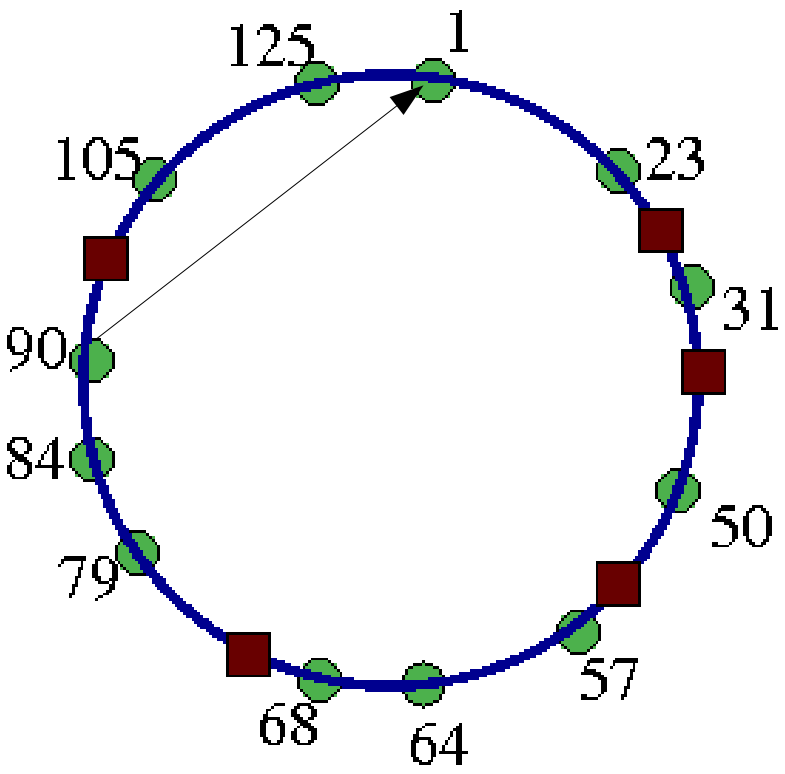


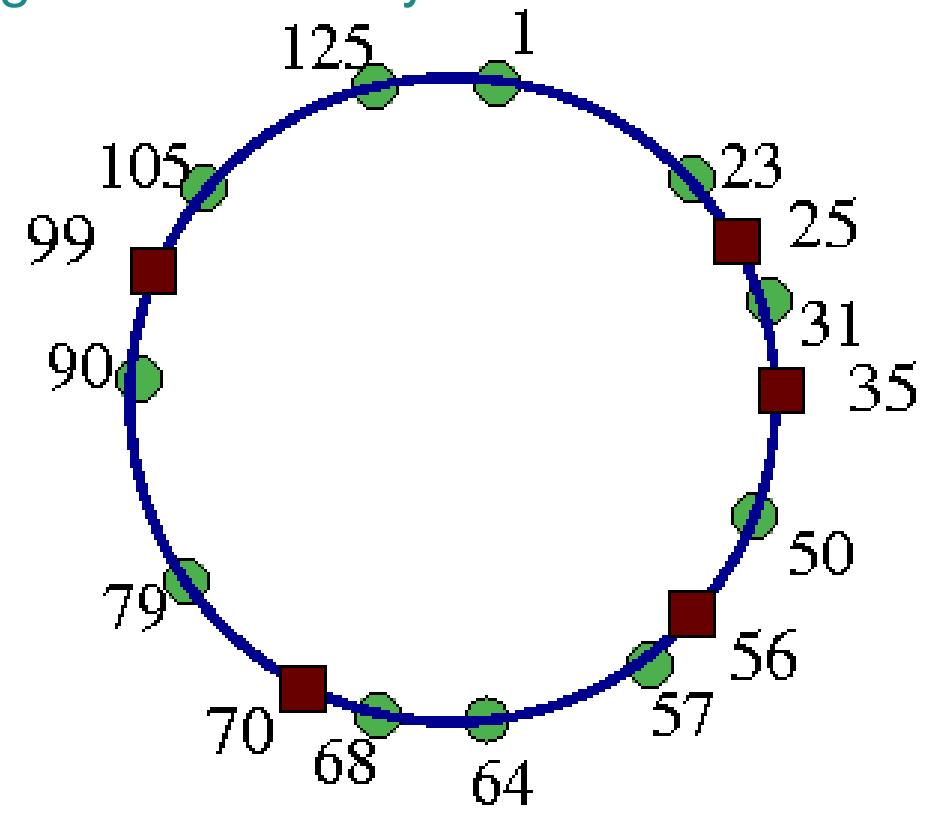


Performance Evaluation

- Churn affects the number of 'wrong' connections in the network
- Wrong connections delay lookups which causes poor performance

Greedy Search on a ring-based Overlay

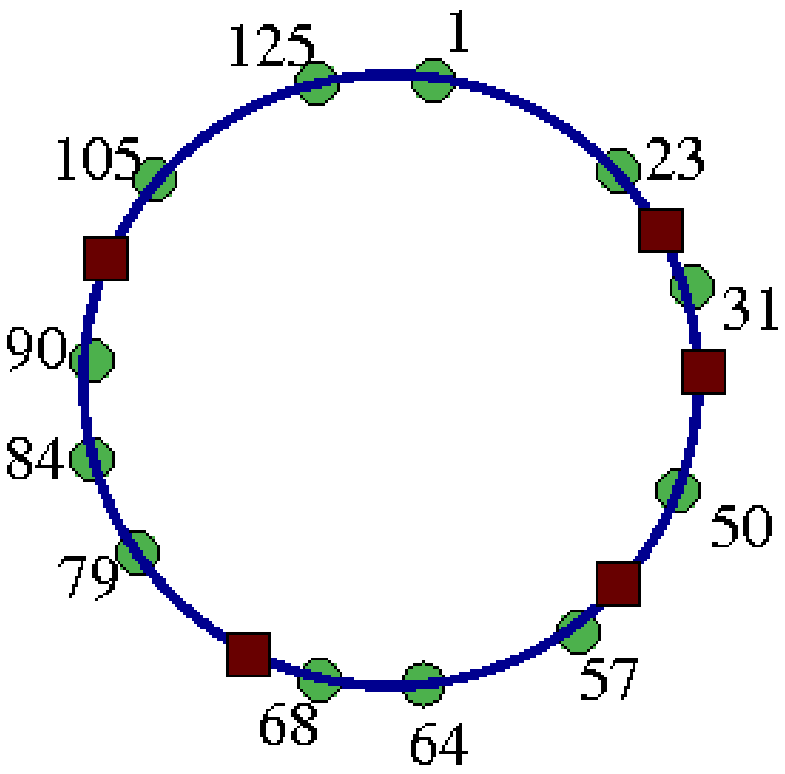


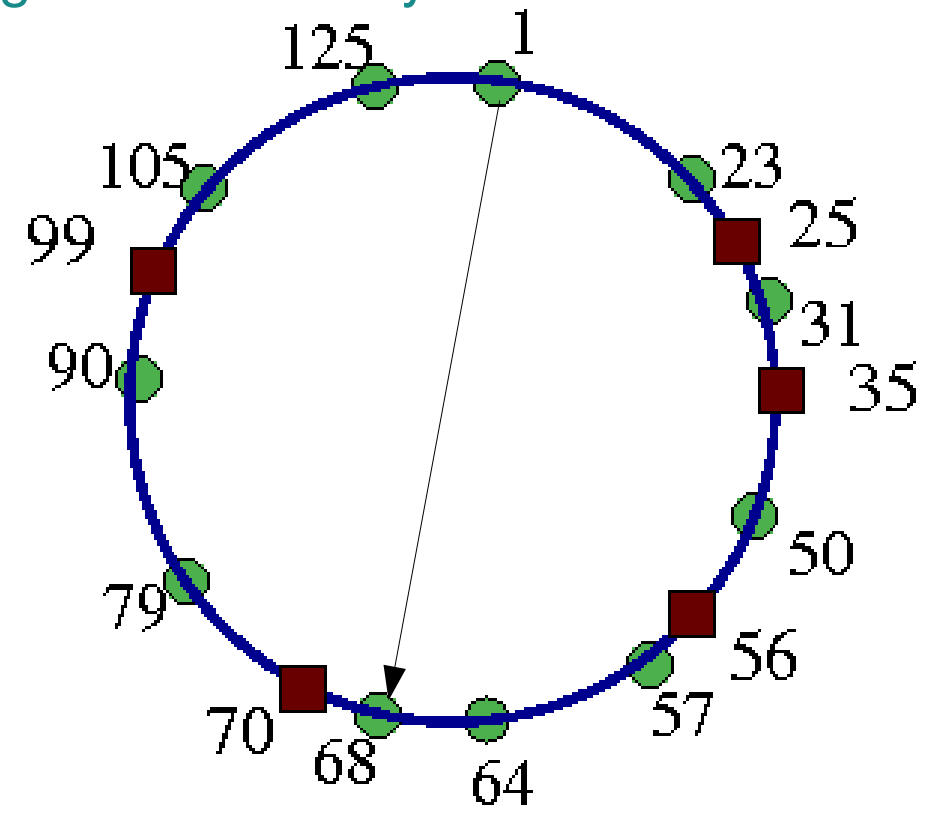


Performance Evaluation

- Churn affects the number of 'wrong' connections in the network
- Wrong connections delay lookups which causes poor performance

Greedy Search on a ring-based Overlay

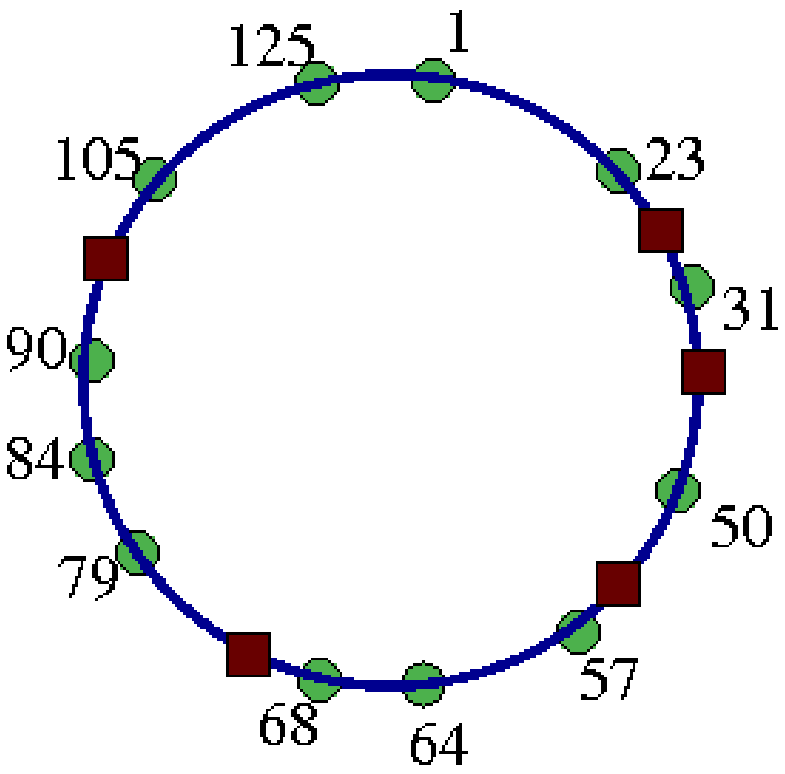


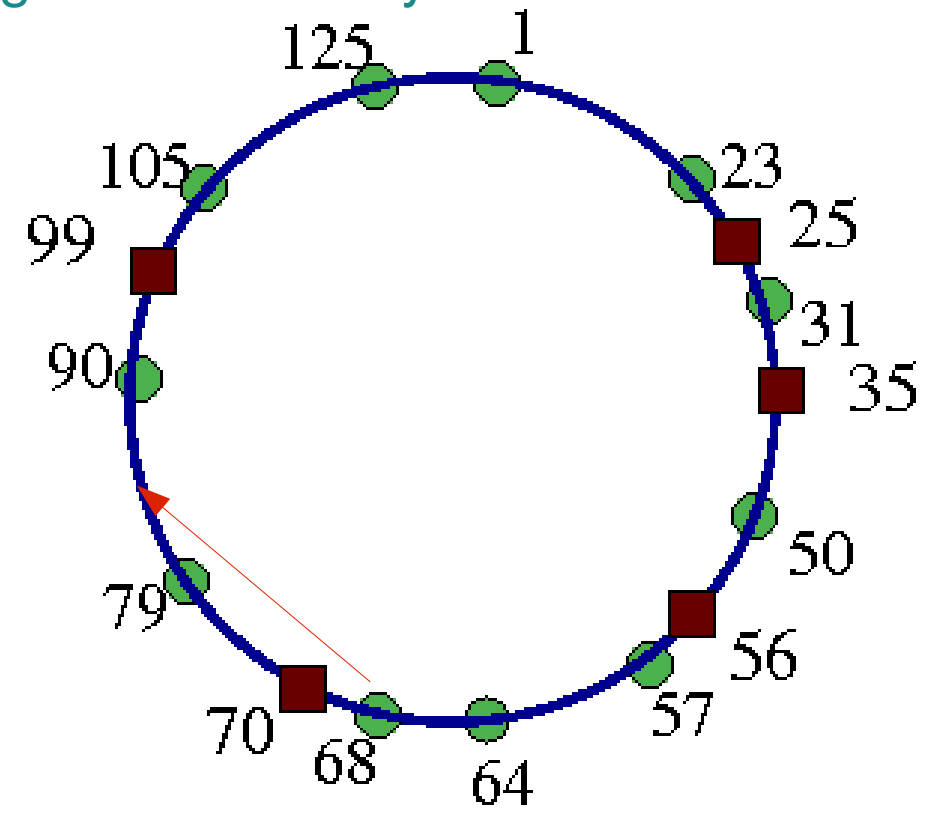


Performance Evaluation

- Churn affects the number of 'wrong' connections in the network
- Wrong connections delay lookups which causes poor performance

Greedy Search on a ring-based Overlay

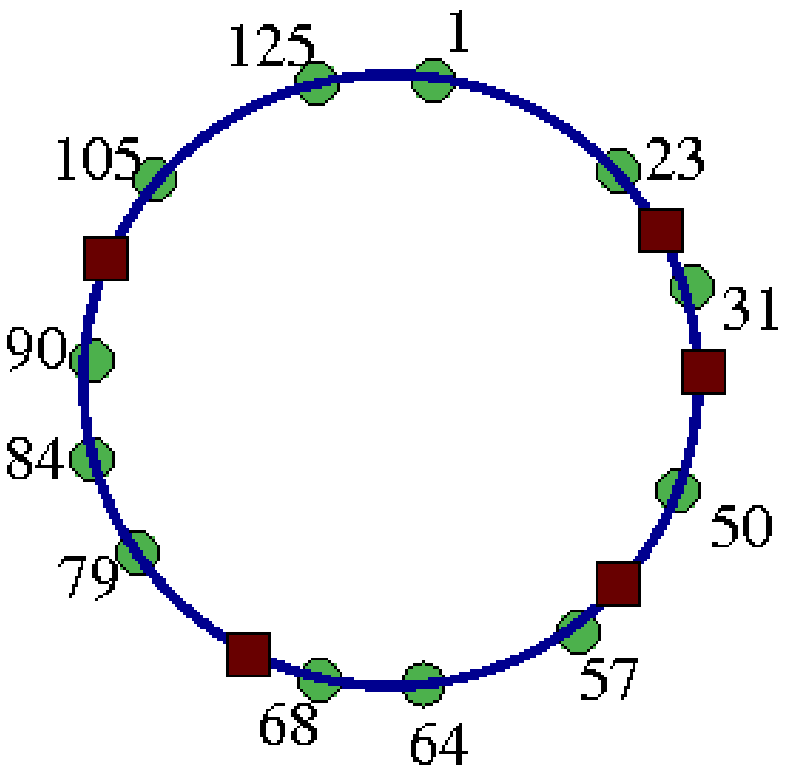


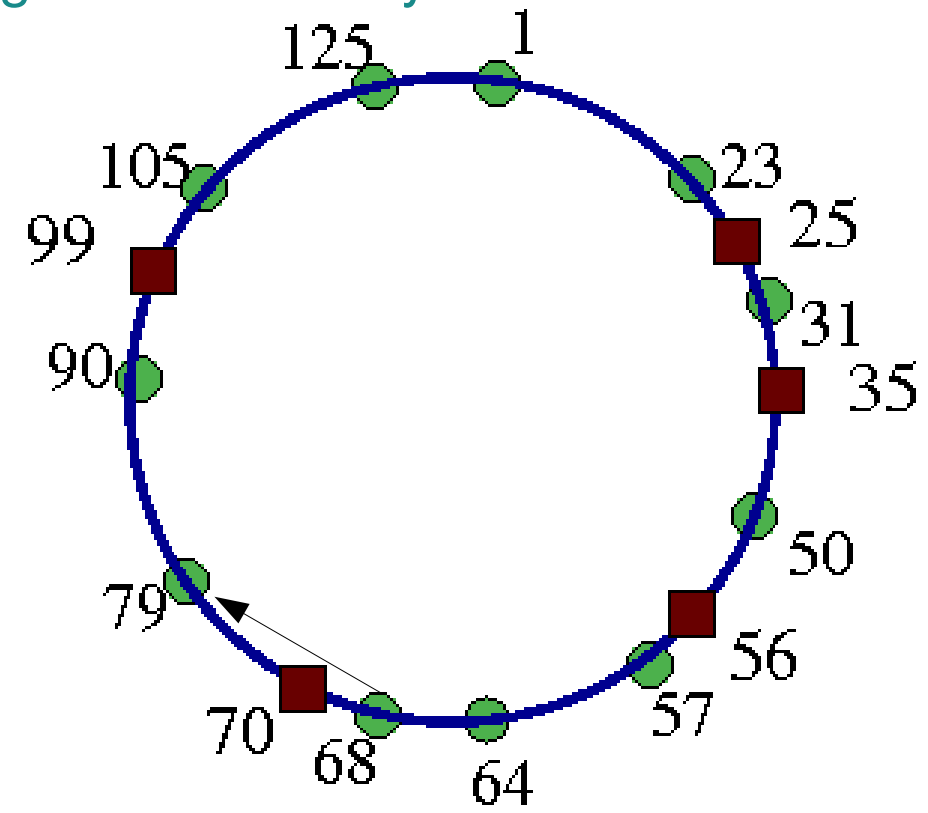


Performance Evaluation

- Churn affects the number of 'wrong' connections in the network
- Wrong connections delay lookups which causes poor performance

Greedy Search on a ring-based Overlay

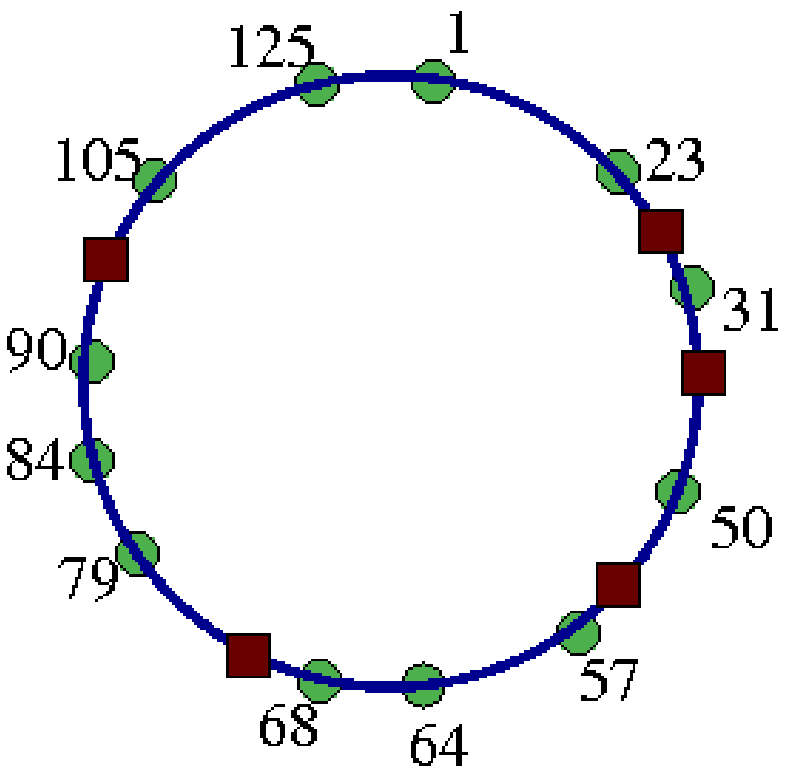


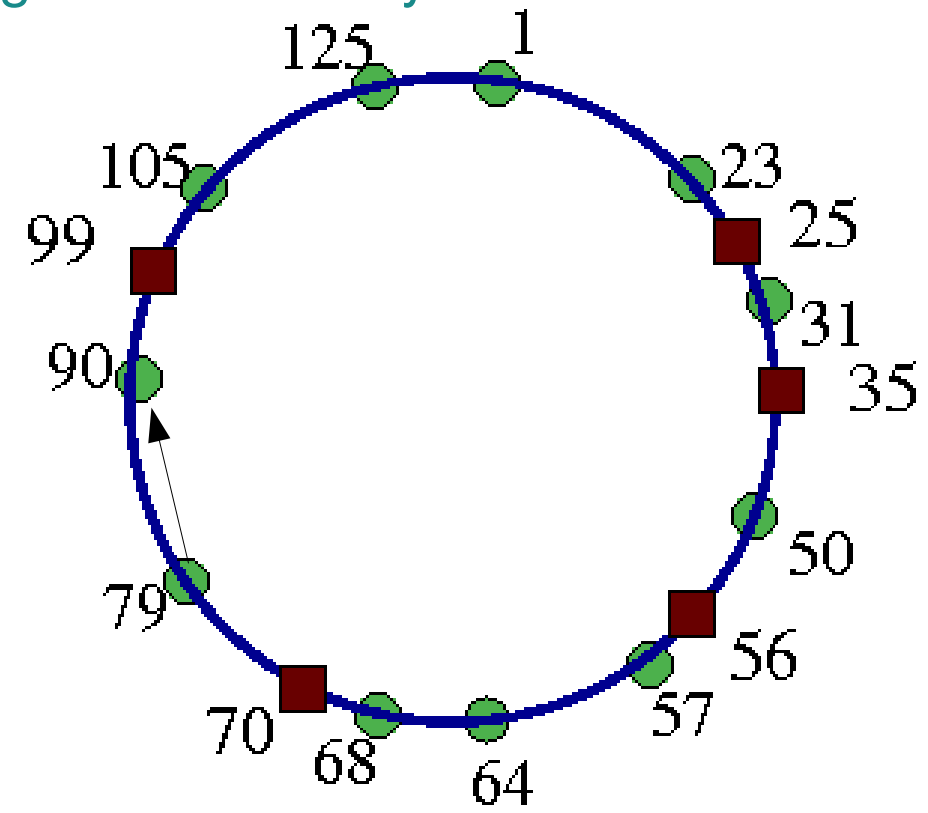


Performance Evaluation

- Churn affects the number of 'wrong' connections in the network
- Wrong connections delay lookups which causes poor performance

Greedy Search on a ring-based Overlay

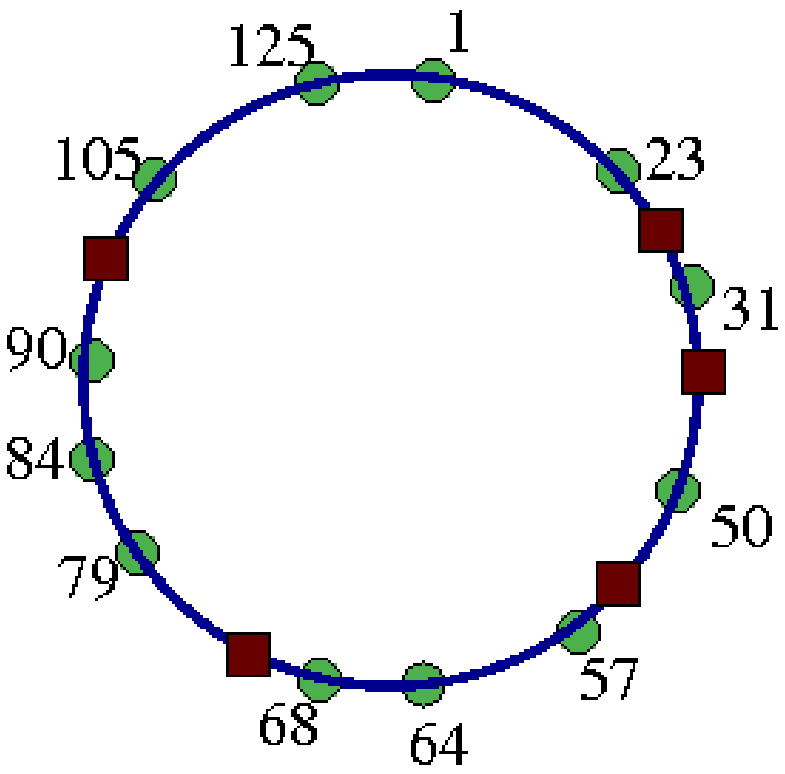


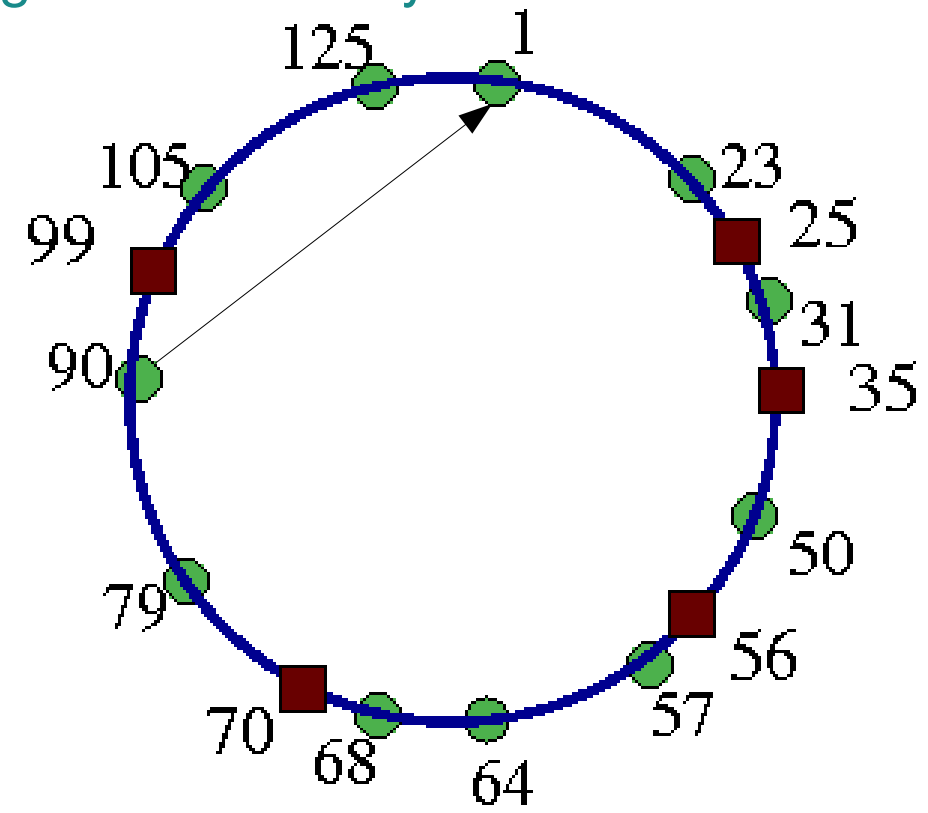


Performance Evaluation

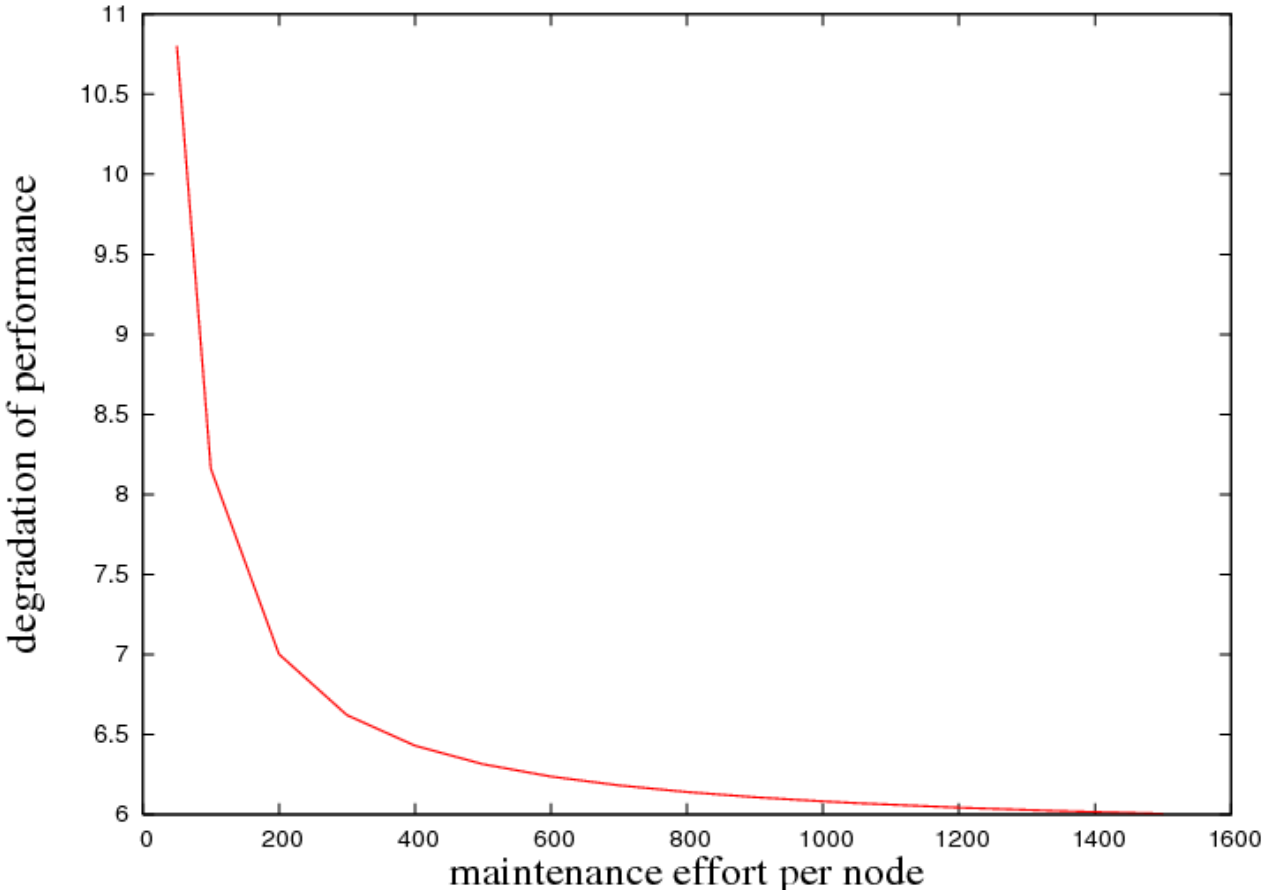
- Churn affects the number of 'wrong' connections in the network
- Wrong connections delay lookups which causes poor performance

Greedy Search on a ring-based Overlay





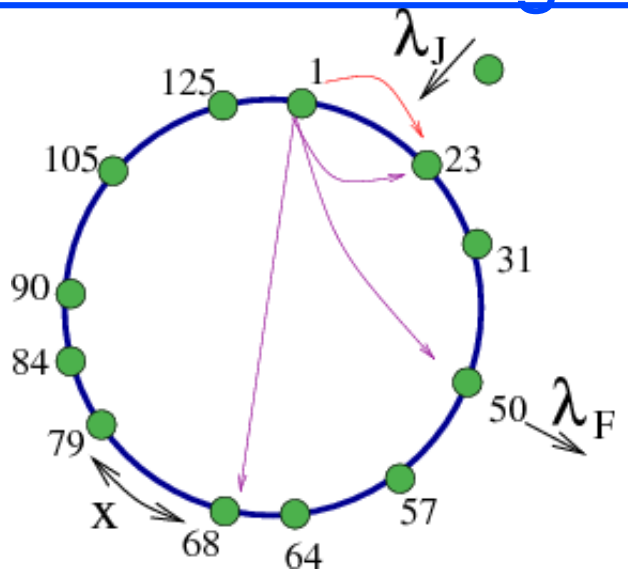
Cost vs. Performance



- Earlier approaches have either been empirical -Li *et al* (IPTPS 2004), Rhea *et al* (Usenix 2004), Castro *et al* (DSN 2004),
- or have focussed on proving theoretical bounds – Liben-Nowell *et al* (PODC 2002), Aspnes *et al* (PODC 2002).

- We are able to predict the performance as a function of the parameters, for the full working range.

Parameters defining the System



K=128
N=12

(S, Fin1,Fin2,Fin3,...Fin7)
= (A,A,A,A,A,A,D,A)

Periodic Maintenance

•Rate of periodically checking a successor or a finger

λ_J : Rate of Joins

λ_F : Rate of Failures

$\lambda_S\alpha$: Rate of checking Successors

$\lambda_S(1-\alpha)$: Rate of checking Fingers

Reactive Maintenance

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

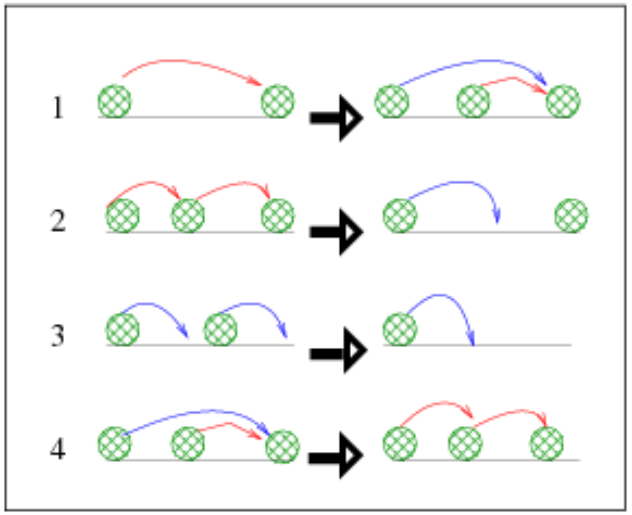
λ_J : Rate of Joins

λ_F : Rate of Failures

λ_S : Rate of checking Successors

λ_M : Rate of sending messages

Probability of Failed or Outdated First Successor Pointers



Change in $W_1(r,\alpha)$	Prob. of Change
$W_1(t+\Delta t) = W_1(t) + 1$	$(\lambda_j N \Delta t)(1 - w_1)$
$W_1(t+\Delta t) = W_1(t) + 1$	$\lambda_f N (1 - w_1)^2 \Delta t$
$W_1(t+\Delta t) = W_1(t) - 1$	$\lambda_f N w_1^2 \Delta t$
$W_1(t+\Delta t) = W_1(t) - 1$	$\alpha \lambda_s N w_1 \Delta t$
$W_1(t+\Delta t) = W_1(t)$	*otherwise*

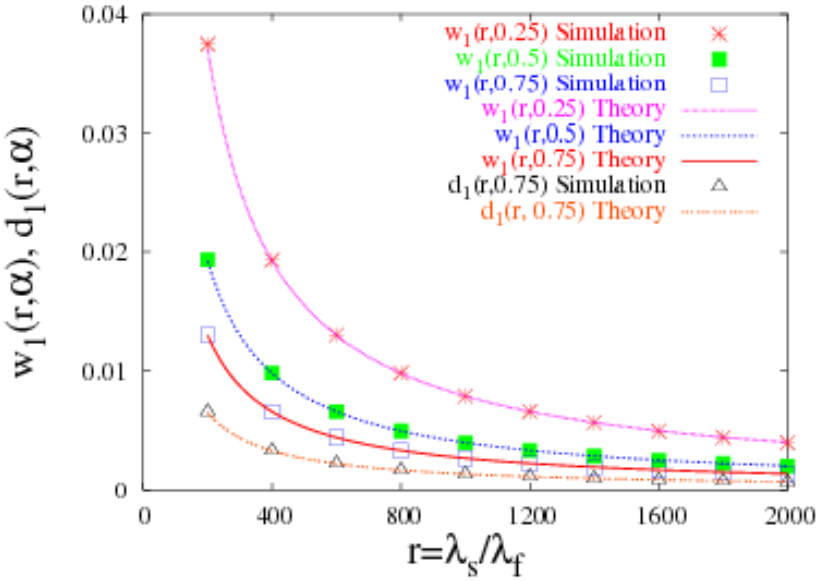
$W_1 \equiv$ the number of wrong (failed or outdated) s_1 pointers

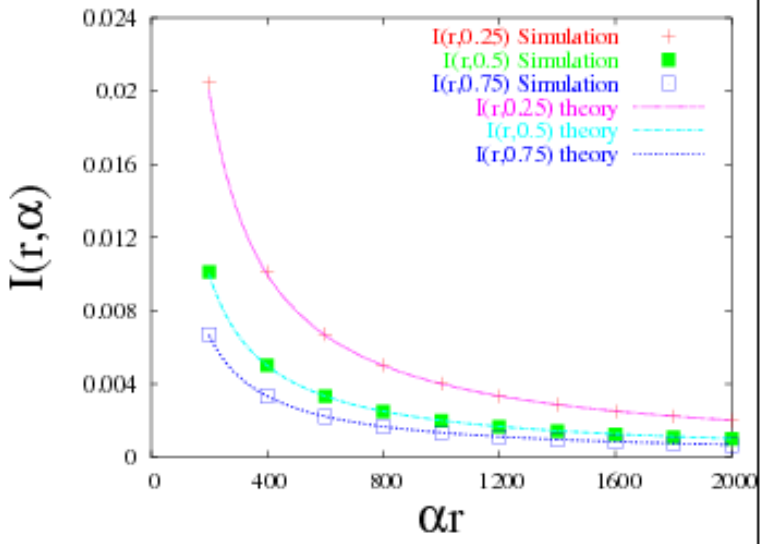
$D_1 \equiv$ number of *failed* s_1 pointers

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

$$w_1(r,\alpha) = \frac{2}{3 + r\alpha} \approx \frac{2}{r\alpha}; \quad d_1(r,\alpha) \sim \frac{1}{2} w_1(r,\alpha)$$

Theory and Simulation for $w_1(r,\alpha)$, $d_1(r,\alpha)$, $I(r,\alpha)$





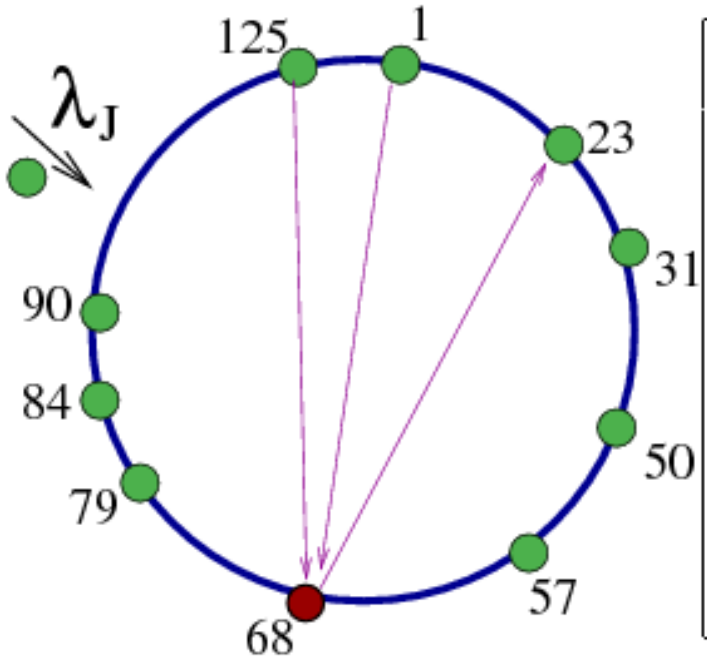
- $w_1(r,\alpha) \equiv$ fraction of incorrect first successor pointers
- $d_1(r,\alpha) \equiv$ fraction of *failed* first successor pointers
- $I(r,\alpha) \equiv$ fraction of lookups which give *inconsistent* answers

$$I(r,\alpha) = w_1(r,\alpha) - d_1(r,\alpha) \sim \frac{1}{\alpha r}$$

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 $n + 2^{k-1}$, $1 \leq k \leq \mathcal{M}$.



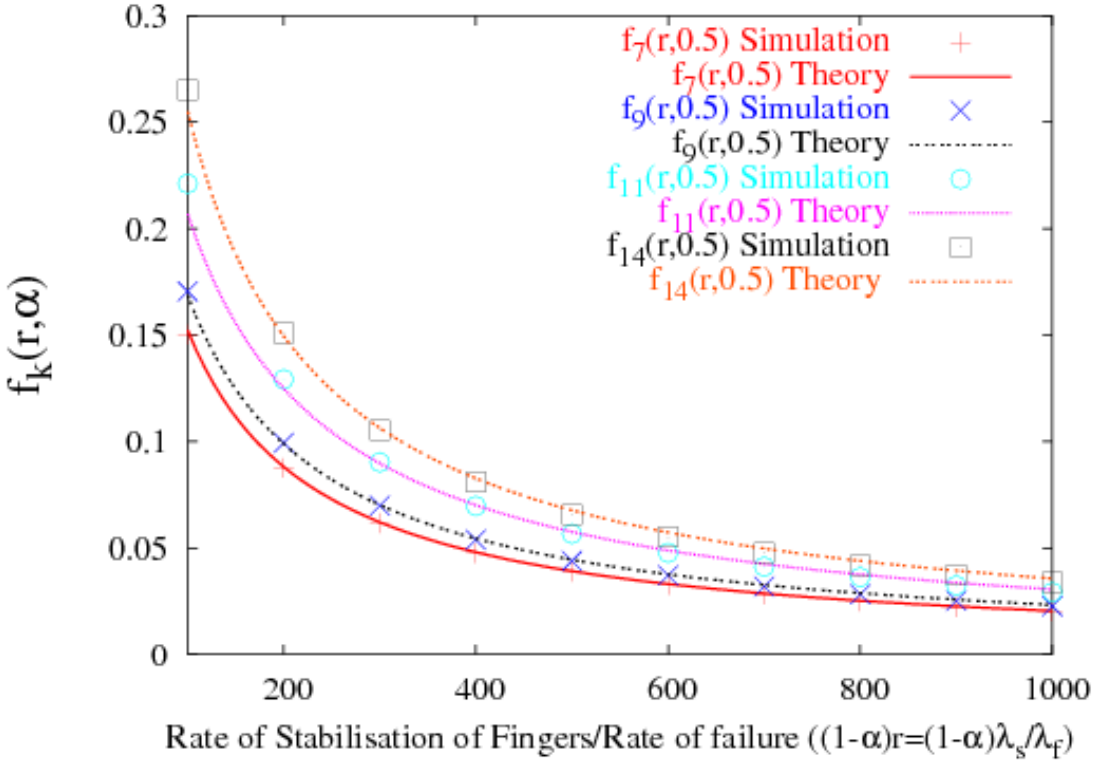
$F_k(t+\Delta t)$	Prob. of Change
$= F_k(t) + 1$	$(\lambda_j N \Delta t) \sum_{i=1}^{k} p_{join}(i,k) f_i$
$= F_k(t) - 1$	$(1-\alpha)\frac{1}{\mathcal{M}} f_k (\lambda_s N \Delta t)$
$= F_k(t) + 1$	$(1-f_k)^2[1-p_1(k)](\lambda_f N \Delta t)$
$= F_k(t) + 2$	$(1-f_k)^2(p_1(k)-p_2(k))(\lambda_f N \Delta t)$
$= F_k(t) + 3$	$(1-f_k)^2(p_2(k)-p_3(k))(\lambda_f N \Delta t)$
$= F_k(t)$	$otherwise$

Number of failed k^{th} Fingers: F_k

(Reactive Maintenance Strategy)

$F_k(t+\Delta t)$	Prob. of Change
$= F_k(t) + 1$	$c_{3.1} = (\lambda_j N \Delta t) \sum_{i=1}^{k} p_{join}(i,k) f_i$
$= F_k(t) - 1$	$c_{3.2} = \frac{f_k}{\sum_k f_k} (\lambda_M N_{S_2} (1 - w_1') A(w_1, w_1') \Delta t)$
$= F_k(t) + 1$	$c_{3.3} = (1 - f_k)^2 [1 - p_1(k)] (\lambda_f N \Delta t)$
$= F_k(t) + 2$	$c_{3.4} = (1 - f_k)^2 (p_1(k) - p_2(k)) (\lambda_f N \Delta t)$
$= F_k(t) + 3$	$c_{3.5} = (1 - f_k)^2 (p_2(k) - p_3(k)) (\lambda_f N \Delta t)$
$= F_k(t)$	$1 - (c_{3.1} + c_{3.2} + c_{3.3} + c_{3.4} + c_{3.5})$

Theory and Simulation for $f_k(r,\alpha)$: The fraction of failed kth fingers



Lookup Hop Count: Theory

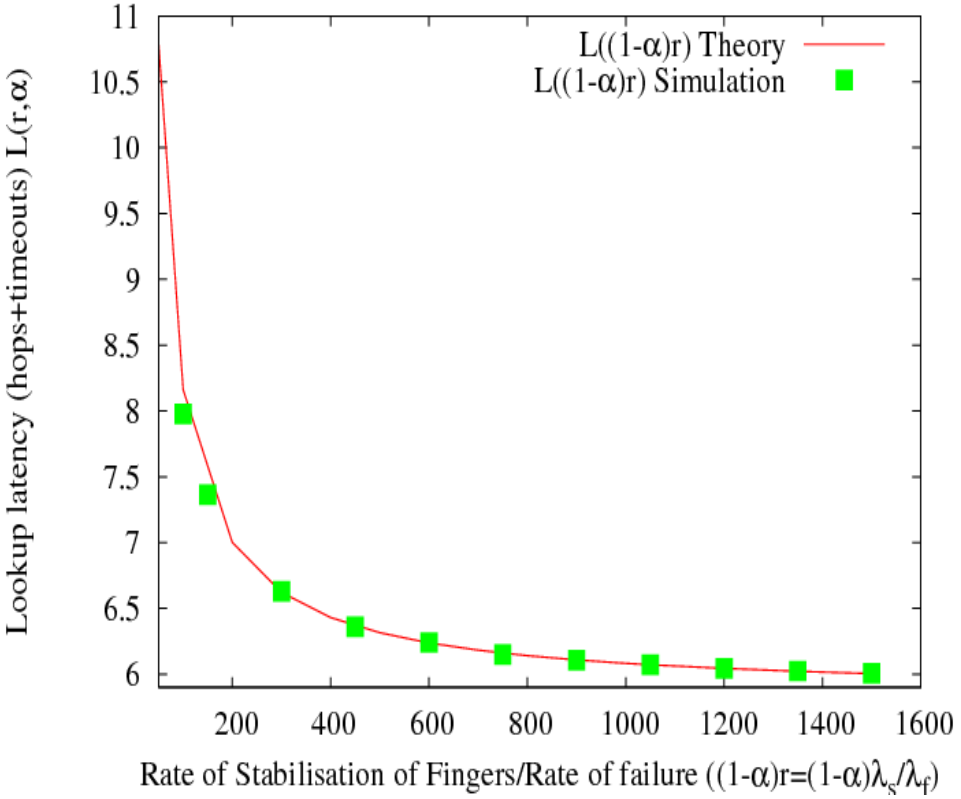
The Cost (Latency) for a node n to lookup a target $t = \xi + m$ is:

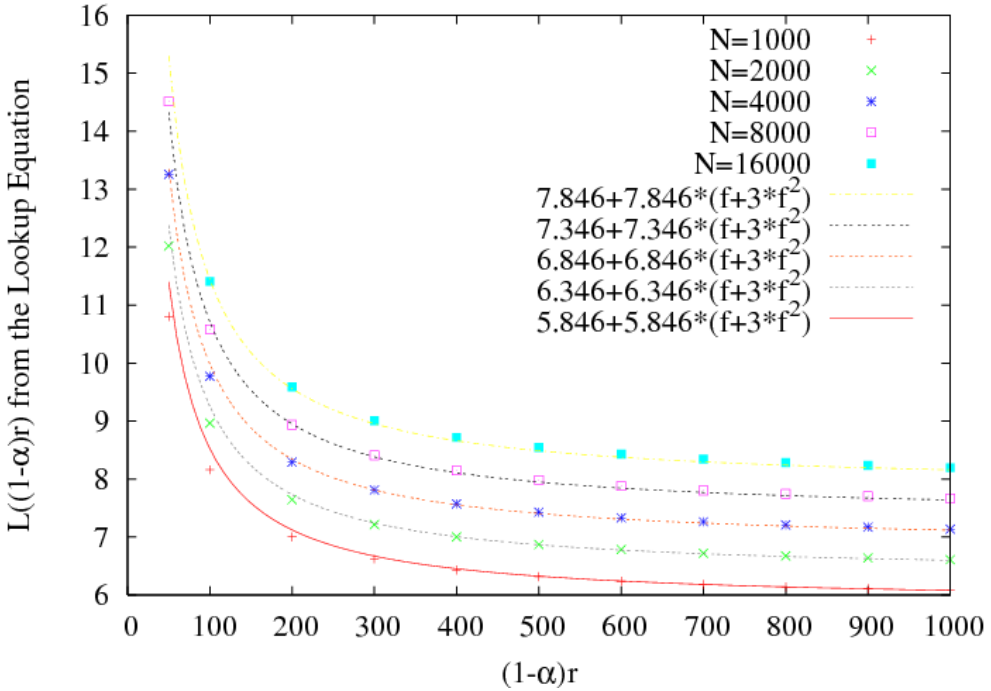
$$
\begin{aligned}
C_{\xi+m} &= C_\xi\,[1 - a(m)] \\
&+ (1 - f_k)a(m)\left[1 + \sum_{i=0}^{m-1} bc(i,m)C_{m-i}\right] \\
&+ f_k a(m)\Bigg[1 + \sum_{i=1}^{k-1} h_k(i) \\
&\qquad \sum_{l=0}^{\xi/2^i-1} bc(l,\xi/2^i)(1 + (i-1) + C_{\xi_i - l + m}) + O(h_k(k))\Bigg]
\end{aligned}
$$

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

$$
\begin{aligned}
h_k(i) &= a(\xi/2^i)(1 - f_{k-i})\Pi_{s=1,i-1}(1 - a(\xi/2^s) + a(\xi/2^s)f_{k-s}) \qquad i < k \\
h_k(k) &= \Pi_{s=1,k-1}(1 - a(\xi/2^s) + a(\xi/2^s)f_{k-s})
\end{aligned}
$$

Results



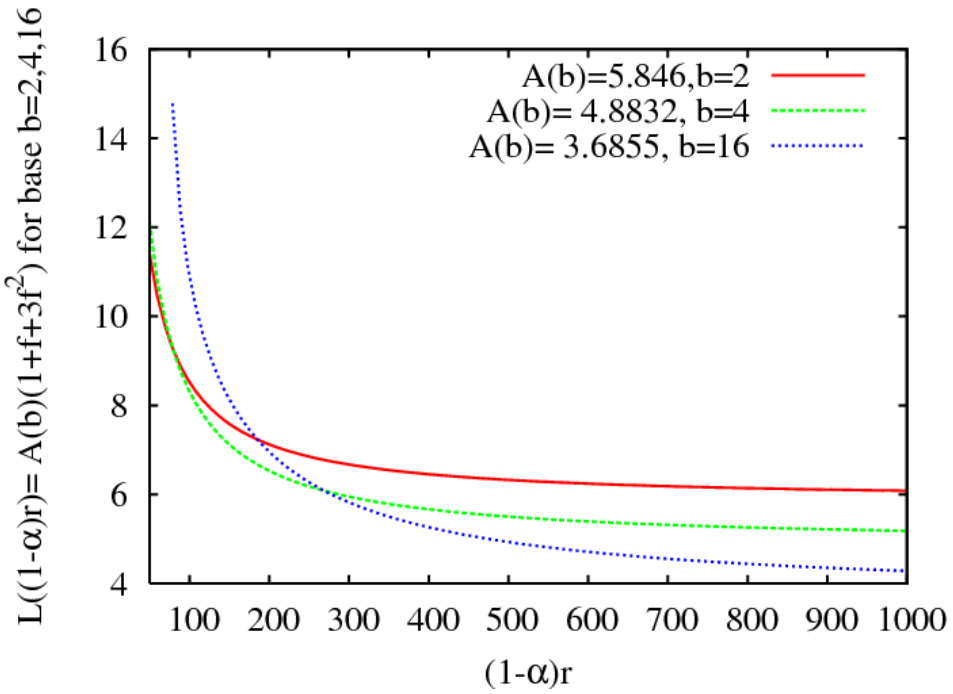


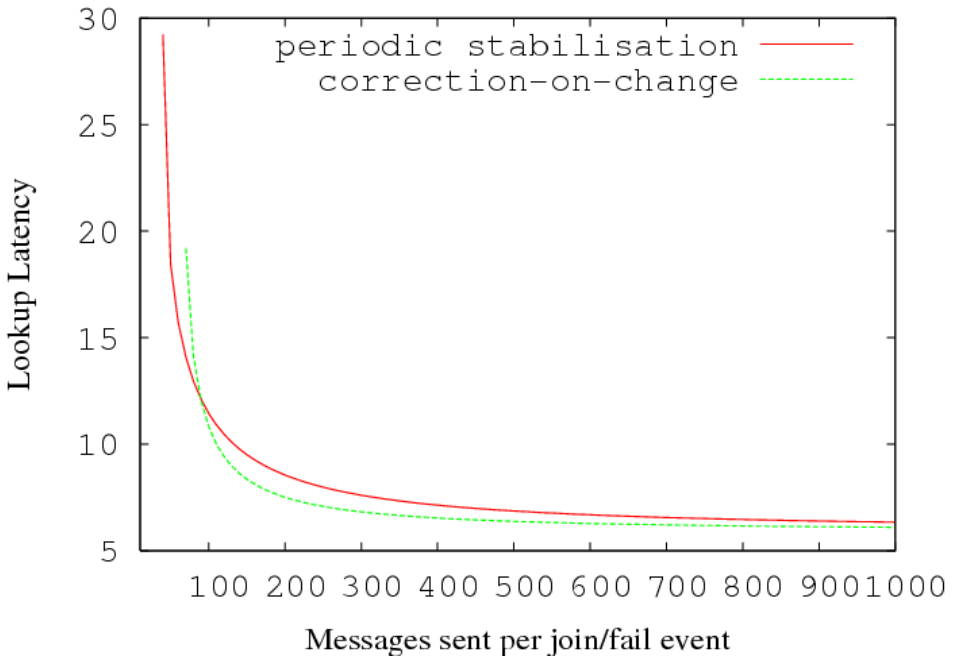
Comparison of theory and Simulations for N=1000

Prediction of Performance for higher N based on functional form

$L \sim \frac{1}{2} \log_2 N(1+f(r) + 3f(r)^2)$

Results(contd.)

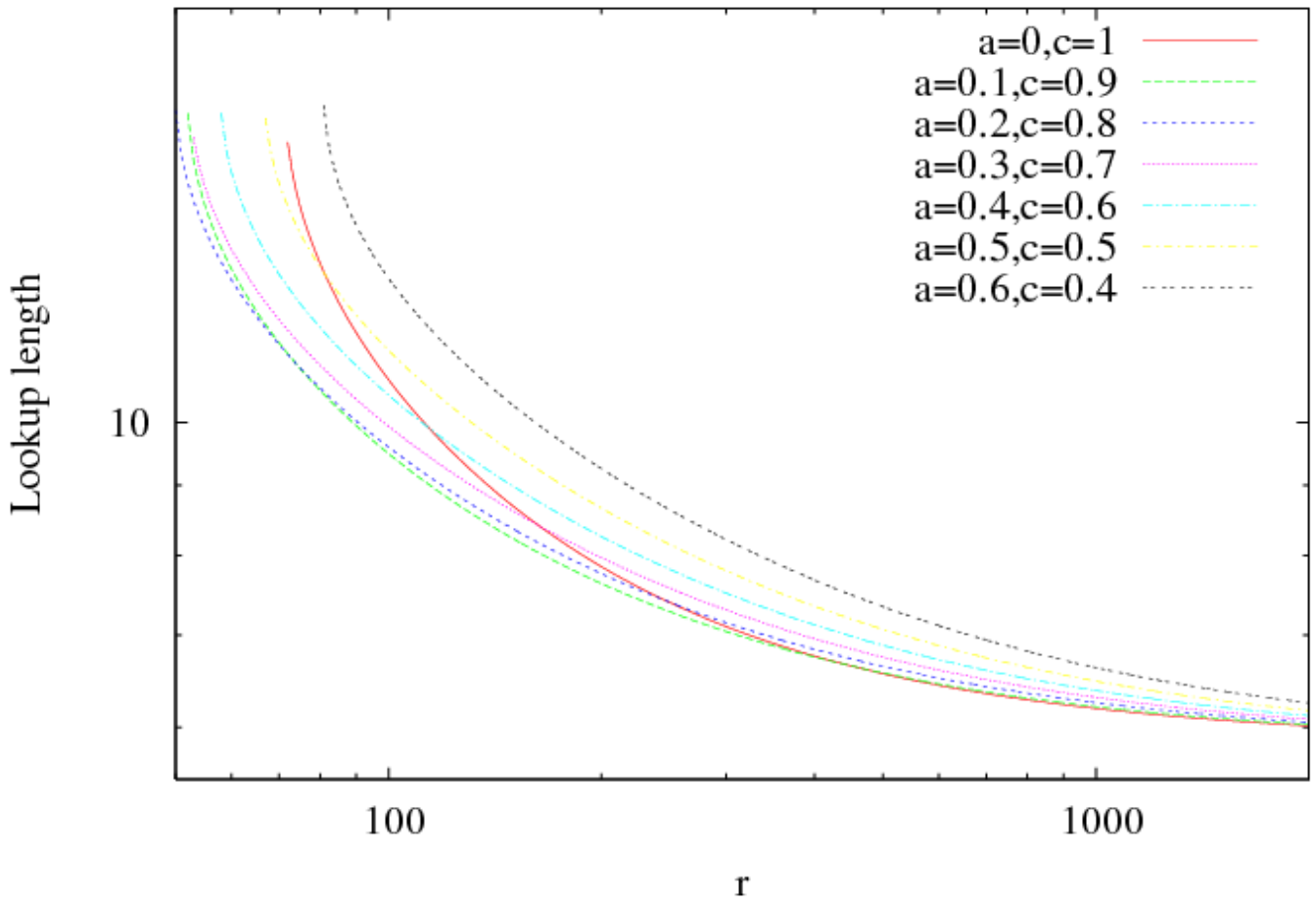




We can predict Performance for different routing table sizes (varying number of connections per node)

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

Results(contd.)



Interestingly, there is a specific value of the control parameter for which the reactive strategy works best

Recent Results and Future Directions

- Lookup *Latency* is a better measure of performance.

- To take the *time* into account, we have generalised the theory so that lookups are not *instantaneous*

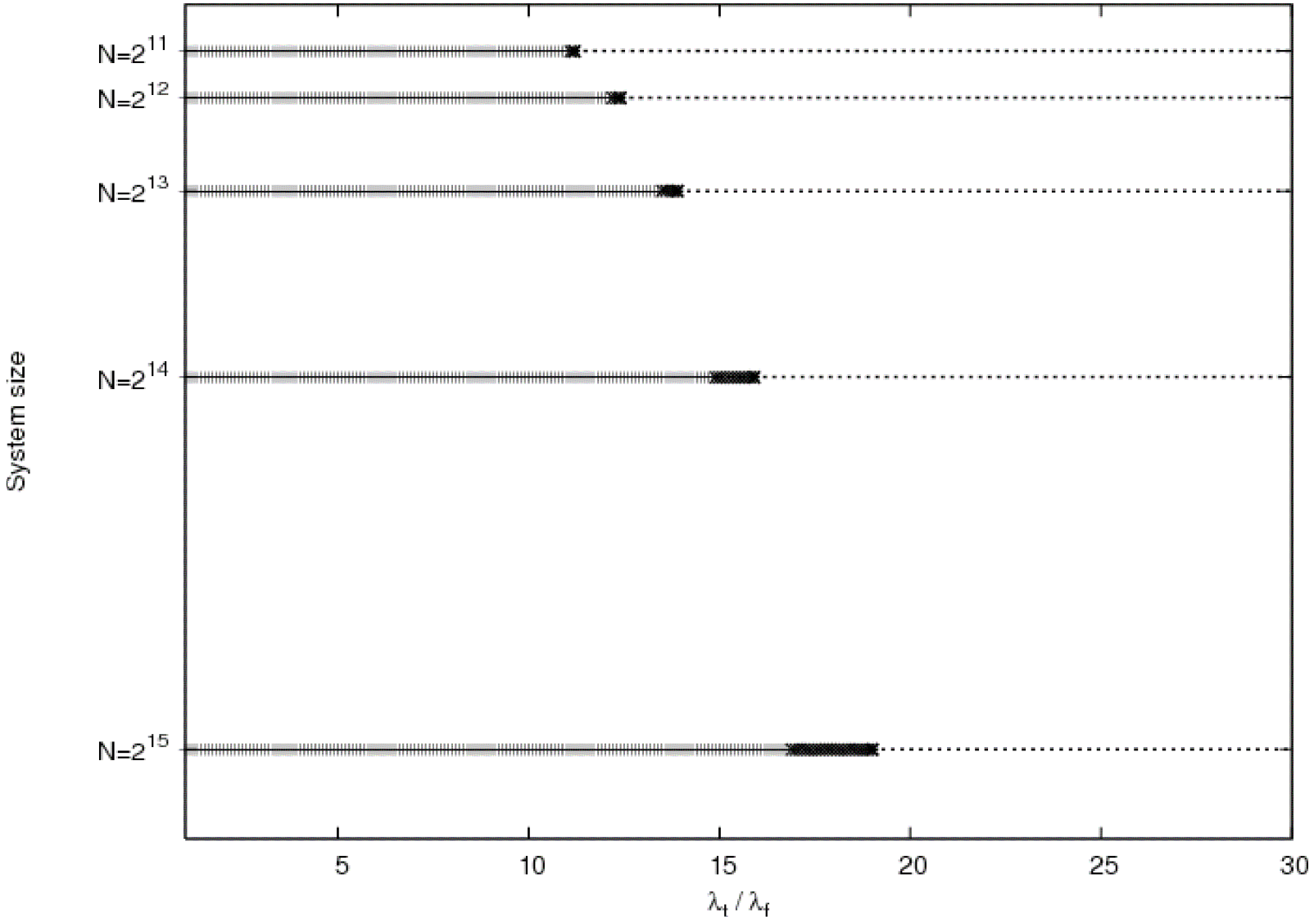
- By doing this, we can also take *proximity* into account

- We analyse two algorithms used by proximity-aware networks (Proximity neighbor selection and Proximity route selection), within the formalism, for a simple distribution of delays, a fraction *p* of the links are slow, and *1-p* are fast.

- For the future, we would like to take *congestion* into account account as well.

Recent Results (contd.)

Phase Transitions in Parameter space



Summary

- We have analysed in detail a structured P2P network and shown that performance can be accurately predicted even while keeping all the details of the functioning of the system.

- We have quantified performance as the number of hops, as well as the *time* taken by an average lookup.

- For the latter case, we need to generalise the theory to take into account the fact that lookups are not instantaneous, due to link- delays.

- New results include the prediction of a region in phase-space where the system does not function efficiently (all the long fingers are dead **all** the time)