# An Empirical Study of the Global Behavior of Structured Overlay Networks as Complex Systems


UCL
Université catholique de Louvain

Ruma R. Paul

KTH Royal Institute of Technology, Sweden
Université catholique de Louvain, Belgium


Licentiate Seminar
October 20, 2015

# Introduction

- Large-scale distributed systems as Complex Systems.
- Many operating modes depending on environment.
- Introduce many problems as stressed beyond where their behavior is a straightforward extrapolation of behavior of their parts.
  - Require continual babysitting.
- Why??

➢ Designed to operate within a specific failure and threat model.

➢ Undefined behavior outside these models.

# What should be the Design Approach?

- *Goal*: Design systems with predictable and reversible behavior in **complete operating space** including highly stressful situations.

- Why it is necessary to explore highly stressful situations?
  - *For practical system design*: Systems running in so- called "stable" environments also have peaks of high stress.
  - *Applications*: New venues for application design, such as mobile and ad-hoc networks and Internet of Things, for which current fault-tolerance techniques are insufficient.
  - *For scientific reasons*: to understand what happens in those regimes.

# Our Approach

- Representative Complex Systems: a class of *Structured Overlay Networks.*

- What is a *highly stressful* or ***inhospitable environment?***
  - Operating environment in which certain stress parameters can potentially reach high values and temporarily increase without bound.

- *For our representative systems we make an empirical study of both the behavior and design in inhospitable environments.*

# Our Approach (cont..)

- Experimental approach
  - Step 1: In Simulated environment (Licentiate Thesis).
  - Step 2: In Real-world dynamic environment, scaling up (Future Work).

- Incremental approach
  - Explore one stress parameter (with zero/constant value of other stress parameters);
  - Investigate interaction between stress parameters.

- Licentiate Thesis: Explored two stress parameters.
  - Churn;
  - Network Partitioning;
  - Interaction between Network Partitioning and Churn.

# Stories and Their Contributions

- **First Story**: "Are the Maintenance Strategies reversible against Churn?"
  - The concept of Knowledge Base is introduced;
  - Establish the necessity of proactive merger using Knowledge Base.

- **Second Story:** "Can we observe the Global Structure? YES! Phase Transitions."
  - Insight on how to observe global structure;
  - Insight on how phase of each node is related to functionality of the system;
  - Empirical demonstration that reversible phase transitions happen in a reversible system as the stress varies.

- **Third Story:** "Can our Maintenance Strategies help with Network Partitioning?"
  - Case of zero Churn: empirical demonstration of power of local maintenance strategies (do not need any explicit merger to achieve reversibility).

- **Fourth Story:** "Understanding how Partitioning and Churn interact"
  - Stranger Model is introduced;
  - Empirical demonstration of need for (proactive) merger (using Knowledge Base).

# Publications

- [1] Ruma R. Paul, Peter Van Roy and Vladimir Vlassov. *An Empirical Study of the Global Behavior of A Structured Overlay Network*. 14th IEEE International Conference on Peer-to-Peer Computing (P2P 2014), September 8-12, 2014, London, England. (Published)

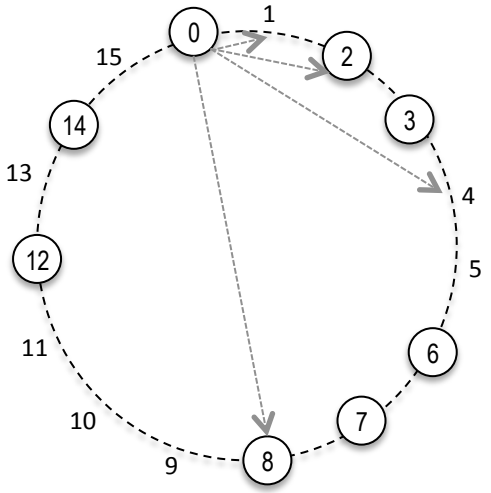
- [2] Ruma R. Paul, Peter Van Roy and Vladimir Vlassov. *Interaction Between Network Partitioning and Churn in a Self-Healing Structured Overlay Network.* 21st IEEE International Conference on Parallel and Distributed Systems (ICPADS 2015), December 14-17, 2015, Melbourne, Australia. (Accepted)

- [3] Ruma R. Paul, Peter Van Roy and Vladimir Vlassov. *Reversible Phase Transitions in a Structured Overlay Network with Churn.* 19th International Conference on Principles of Distributed Systems (OPODIS 2015), December 14-17, 2015, Rennes, France. (Submitted)

# Representative Complex Systems: Ring Structured Overlay Networks

# Design Aspects[1] & An Example Ring Overlay (Chord)

- Choice of Identifier Space: A subset of natural numbers of size N, with closeness metric, d(x, y) = (y − x) mod N .
  - ✓ Example: Space N=16 {0,…,15}; d(8,14)=6 and d(14,6)=8
- Mapping to the Identifier Space: A uniform hash function or some random function.
  - ✓ Example: A uniform Hash function. H(a)=7, H(d)=14
- Management of the Identifier Space: A peer with virtual identifier p is responsible for the interval (predecessor(p),p].
  - ✓ Example: Peer with virtual id 12 is responsible for (8,12] ({9,10,11,12}).
- Graph Embedding: Belongs to "routing-efficient" small-world networks.
  - ✓ Example:
    - Each peer views the identifier space as partitioned in $\log_2$ (N ) partitions where each partition is 2 times bigger than the previous one.
    - Each peer has a routing table which contains $\log_2$ (N ) links to some nodes from each partition.
    - *Routing_Table*(0) = {2,6,8}.
- Routing Strategy: Greedy routing strategy.
- Maintenance Strategy: Required to maintain the structural integrity as peers join, fail or leave.
  - ✓ Example: Periodic Stabilization.






[1]Aberer et. al., "The essence of P2P: a reference architecture for overlay networks," in *Proc. P2P*, 2005.

# Ring Overlays (Cont..)

- Many variations of Chord:
  - To gradually improve and circumvent or relax the requirements of a perfect ring for accuracy.
  - Join in Chord may be incorrect (inconsistency due to concurrent joins).
  - DKS: correct join, but requires simultaneous locking of multiple nodes.
  - Beernet: correct lock-free two step join.

# Stress in Operating Environment

- What are the *stress parameters*?
  - Identified and organized operating space using 5 parameters:
    - Churn: node turnover.
    - Network Partition: partition of the system.
    - Network Dynamicity: changes of connectivity.
    - Workload: workload beyond capacity.
    - System Size: measurement of scalability.

# Self-* Properties

- To survive inhospitable environments, Self-* properties are crucial.
  - In particular requires **complete self-healing**.

- Structured Overlay Networks (***SONs***) provides certain self-* properties.
  - Existing Maintenance in SONs shows partial self-healing behavior.
  - Existing literature lacks assessment or verification of SONs reversibility by complete self-healing.

# Reversibility & Self-Stabilization

- ***Reversibility*** Property of a system:
  - Ability to repair itself to provide its original *functionality* when the external stress is withdrawn.
  - Functionality of a system is a property of current environment hostility and not of the history of environment hostility.

- ***Self-Stabilization*** and ***Reversibility*** :
  - Self-stabilization talks about system state (internal); Reversibility talks about system functionality (behavior, external).
  - A self-stable system is able to survive arbitrarily high levels of transient failures (state perturbation); Reversibility in a SON might also concerns about permanent failures, Churn.
  - Self-stabilization means resilient to arbitrary temporary failures; Reversibility in case of SON is different – combination of permanent node failures and temporary communication failures:
    - Churn: Nodes fail (permanently) and new nodes arrive to replace them. At system level, this is like a temporary perturbation. At node level, there are only permanent failures.
    - Partitioning: It is temporary but it is external to the nodes – communication problems, no perturbation of system state.

# Maintenance Strategies

- Goal of this work: enhancement of maintenance in our representative complex systems to survive in inhospitable environments.

| | Reactive | Proactive |
|---|---|---|
| **Local** | Correction-on-Change (for self-healing) and Correction-on-Use (provides self-optimization and self-configuration).<br>Ex. DKS, Beernet. | Periodic Stabilization: correction using periodic probing.<br>Ex. Chord, Chord#. |
| **Global** | Overlay Merger with Passive List: Trigger Merger using falsely suspected nodes[2]. | **Overlay Merger[2] with Knowledge Base: Proactive approach to trigger merger using the gathered knowledge at each node (our contribution).** |



. [2]T. M. Shafaat, "Partition tolerance and data consistency in structured overlay networks," Ph.D. dissertation, KTH, Sweden, 2013.

# Knowledge Base (KB)

- Best-effort view of global membership of the system at each node.
- Each node keeps on learning about other nodes, build an "acquaintance" list.
- Different levels of KB:
  - Passive KB: Built through listening only.
    - At peer *p*: for each new node *a*, *p* comes across (while routing or as a member of its current neighborhood), virtual id and network reference of *a* is added to $KB_p$.
    - Not shared with other peers, thus no bandwidth consumption.
    - May be out of date quicker.
  - Active KB: a node communicates with others to enhance its KB.
    - A weak algorithm, e.g. each joining node informs others;
    - A stronger algorithm, such as gossip where each node asks a random node to send its KB that it unions with its own KB.
    - Faster convergence of KB.
    - Extra Bandwidth consumption.
  - Oracle: information from "outside the system".
    - Required when system is unable to achieve reversibility by itself.

# Maintenance Strategies (Cont..)

- Proactive Merger using Knowledge Base: Each node optimistically samples in a periodic manner from its KB to trigger merger.

- **KB** and **Gossip Frameworks**:
  - Gossip can be used to build Knowledge Base – only in case of Active Knowledge Base.
  - *Future work*: improved maintenance and application of knowledge, by borrowing ideas from gossip protocols.

- *Are these the only possible maintenance strategies?*
  - Probably not!! But they cover a large part of the operating space;
  - They have proven their usefulness in practice (we have extended them when we found gaps, like KB);
  - Other strategies exist, e.g., gossip, with different properties, some of our future work will investigate this.

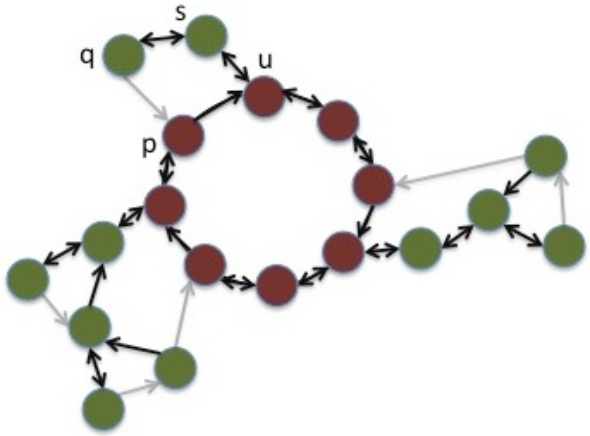
# Summarizing Maintenance Strategies

| Principles | Local/ Global | Reactive/ Proactive | Fast/Slow | Safety | Bandwidth Consumption |
|---|---|---|---|---|---|
| Correction-on-* | Local | Reactive | Fast | Yes | Small |
| Periodic Stabilization | Local | Proactive | Slow | Lookup inconsistencies and uncorrected false suspicions can be introduced | High |
| Merger with Passive List | Global | Reactive | Adaptable | Yes | Adaptable |
| Merger with KB | Global | Proactive | Adaptable | Yes | Adaptable |

# Beernet

- Beernet[3] is a representative example of the design class as per the reference architecture proposed by Aberer et.al.

- Why Beernet?
  - Has properties typical of our design class. Similar to Chord, but has a correct lock-free join operation.
  - Relaxes ring membership operations.
    - Requires only simple message passing, never locking of multiple nodes.
  - Based on 2 invariants.
    - *Every peer is in the same ring as its successor;*
    - *A peer does not need to have connection with its predecessor, but it must know its predecessor's key.*
  - Consequence: Natural Branching structure. A stable **core ring** and transient **branches**.
  - Nodes join in two steps: first, on branch, then on the core ring.




Branches on a relaxed ring. Peers p and s consider u as successor, but u only considers s as predecessor. Peer q has not established a connection with its predecessor p yet.

[3]B. Mejías, "Beernet: A relaxed approach to the design of scalable systems with self-managing behaviour and transactional robust storage," Ph.D. dissertation, UCL, Belgium, 2010.

# Evaluation

- An overlay of 100 peers.
- All experiments are done in Mozart-Oz 2.0 in a simulated environment.
- Experiments were done to observe the effects of the maintenance techniques in various parts of the operating space (high values of stress parameters).
- Ongoing Work: Experimentation in a real-world dynamic environment (not simulated).

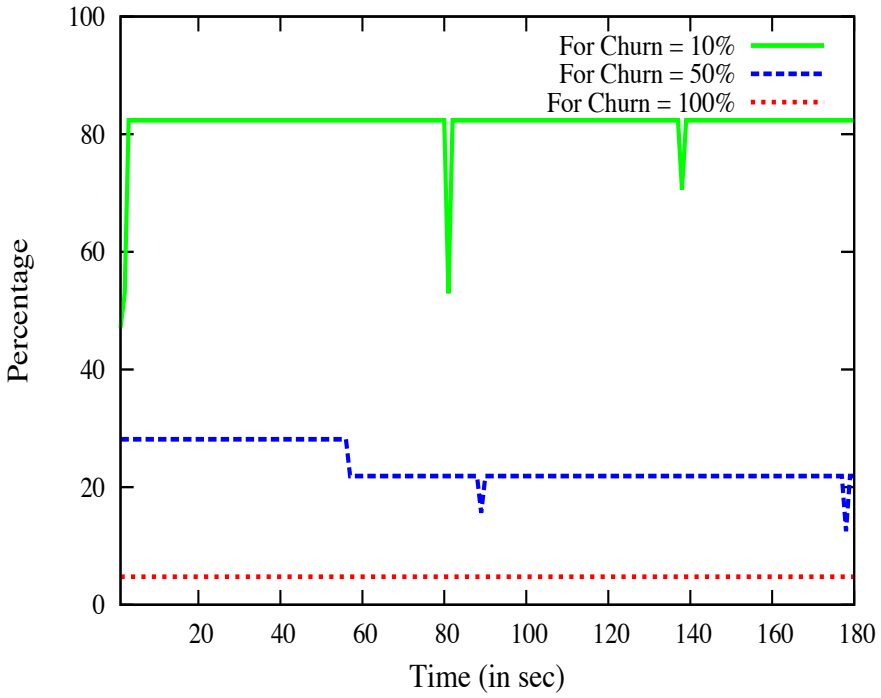
# First Story
# Churn & Reversibility

# Are the Maintenance Strategies Reversible? (1)

Churn: % of node turnover per second.

Metric: % of nodes on core ring as a function of time

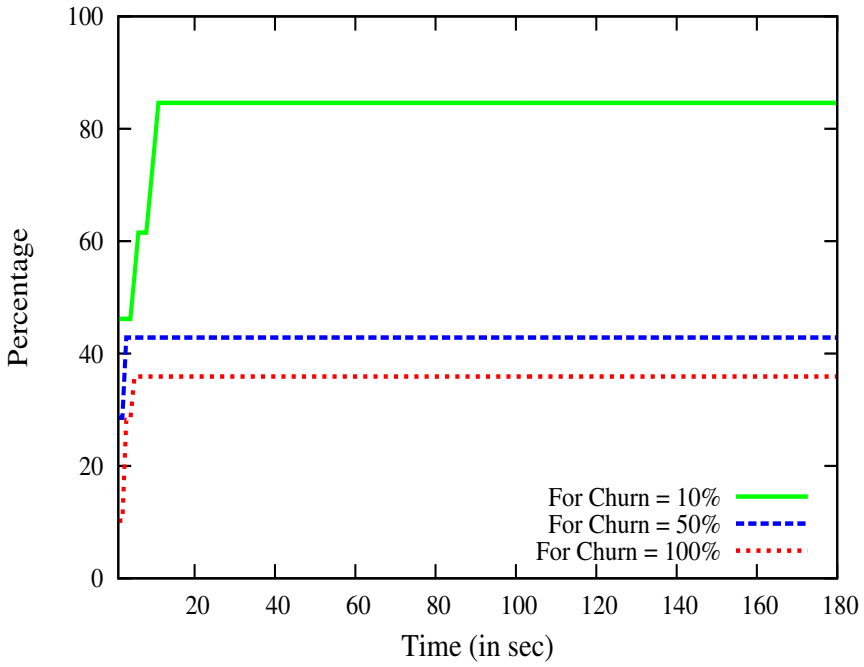
Correction-on-*



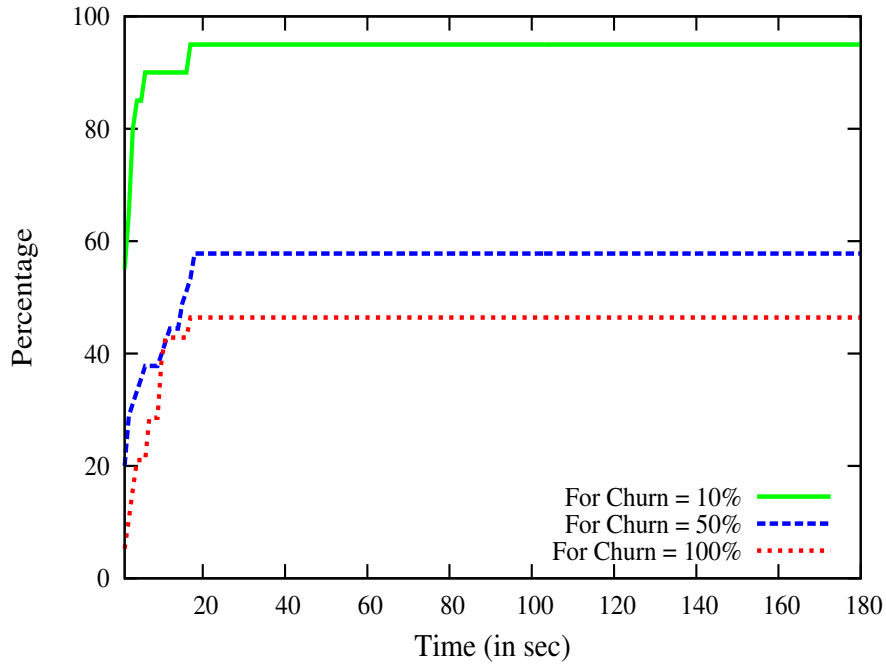

Discontinuation of Self-healing due to lack of liveness!!

# Are the Maintenance Strategies Reversible? (2)

Correction-on-* and Periodic Stabilization




Correction-on-* and Periodic Stabilization and Merger with passive list

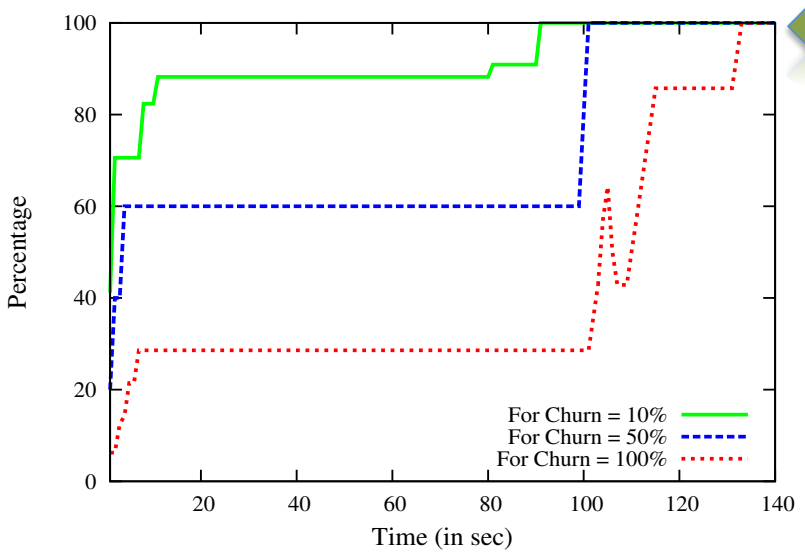



**Still not Reversible**. Why?

# Are the Maintenance Strategies Reversible? (3)

- Incomplete/pending joining of new nodes.
- High churn makes overlay unstable, which do not allow new peers.
  - Rapidly invalidates the join reference new peer has.
- In order to make these isolated peers part of overlay, need to re-trigger join by providing new valid join reference.
  - **Knowledge Base** is required to get knowledge about an alive peer of overlay.
- **Proactive triggering of merger using Knowledge Base** to avoid partition of the system after isolated nodes complete their join procedures.

Correction-on-*, Periodic Stabilization,
Merger with Knowledge Base.




A Perfect Ring with 100% nodes!!

# Second Story
# Phase Transitions

# Phase, Phase Transition & Critical Point

- System = An aggregate entity composed of a large number of interacting parts.
  - Each part is a node of the SON.

- A *Phase* is a subset of a system for which the qualitative properties are essentially the same.
  - Different parts can be in different phases, depending on the local environment observed by the part.

- Why is this interesting?
  - System functionality depends on these qualitative properties ;
    - Use phase for observing system functionality, but it should work without extra computation and even when communication is broken;
  - Important for applications running on top of SON in a hostile setting.

# Phase, Phase Transition & Critical Point (Cont..)

- A *Phase Transition* occurs when a significant fraction of a system's parts changes phase.
  - This can happen if the local environment changes at many parts.

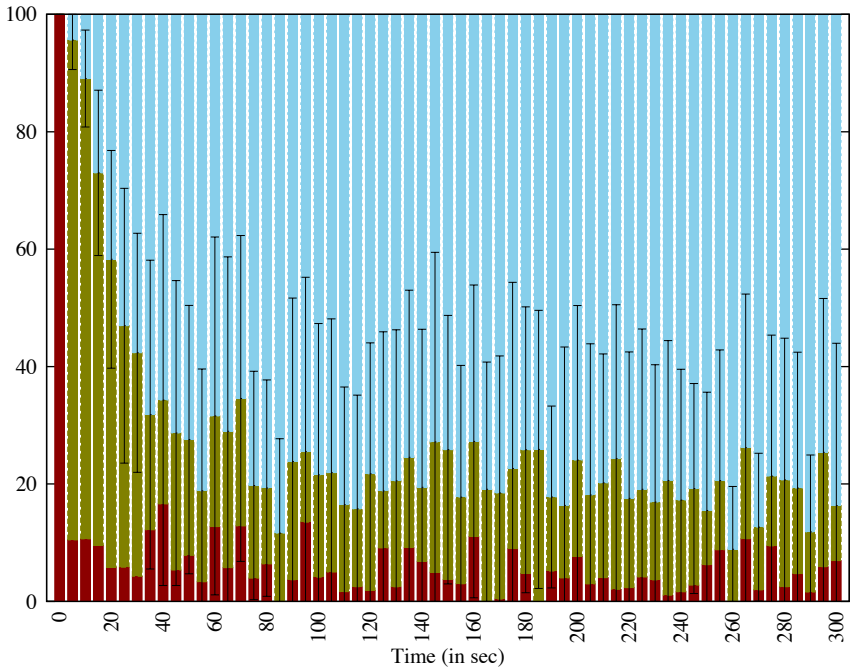
- A *Critical Point* occurs when more than one phase exists simultaneously in significant fractions of a system.

# Can we observe the global structure? YES! Phase transitions !!

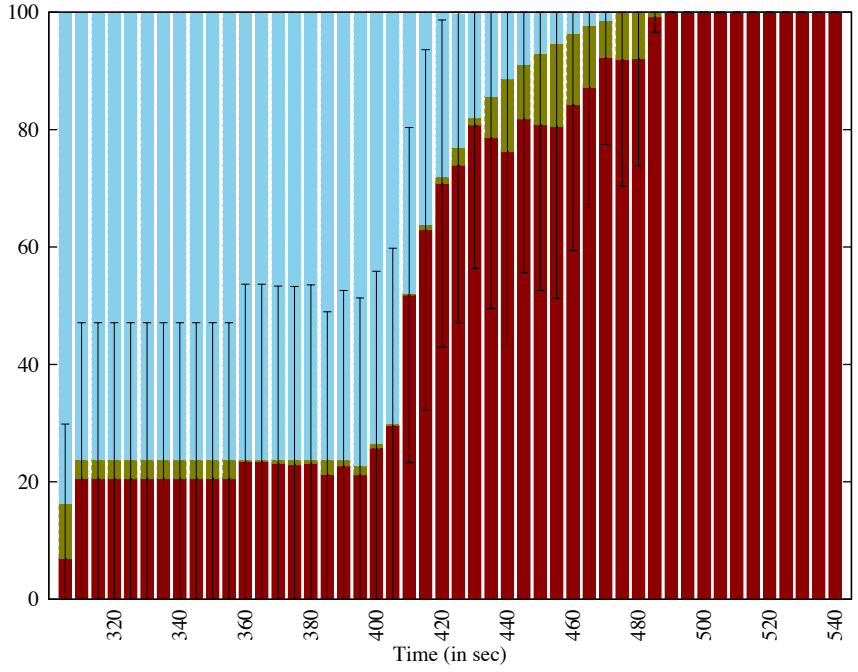
- Phase transitions between *solid*, *liquid* and *gaseous* states.
  - *Solid*: neighbors do not change (core ring).
  - *Liquid*: neighbors changing (branches).
  - *Gaseous*: no neighbors (isolated nodes).

- Phase transitions are not global synchronizations.
  - At one time, each node can be in a different phase,
  - Different phases can coexist in the same system.

Phase Transitions in SON: red, green and blue areas correspond to % of nodes on ring (*solid*), branches (*liquid*) and isolation (*gaseous*) respectively.

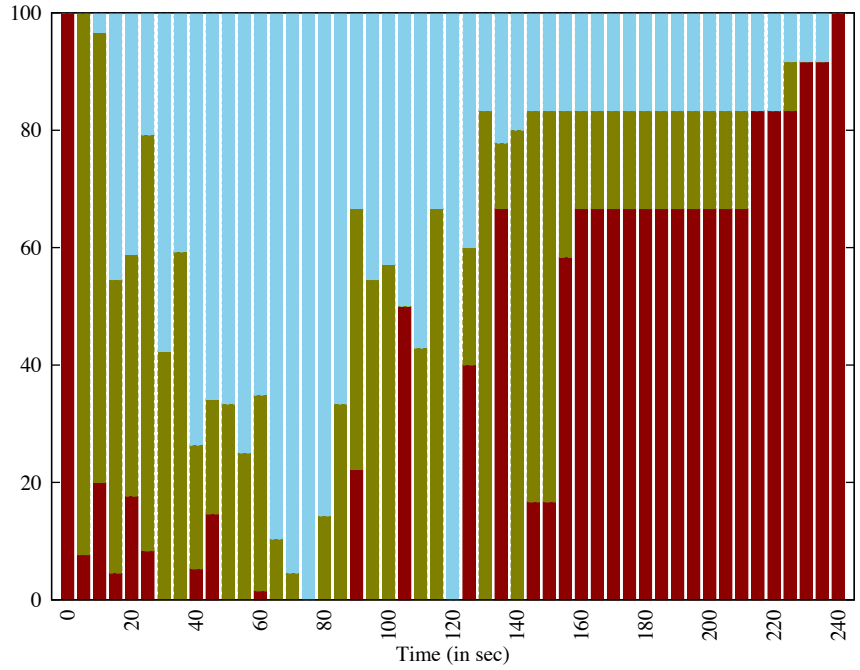



Under increasing churn during 5 minutes




After withdrawing churn

Increasing churn with time up to a gaseous state, then decreasing churn with time:




What are Phase Transitions good for?

- ✓ Give useful information to the application.
- ✓ Can be observed easily (by-product of maintenance, no global sync needed)

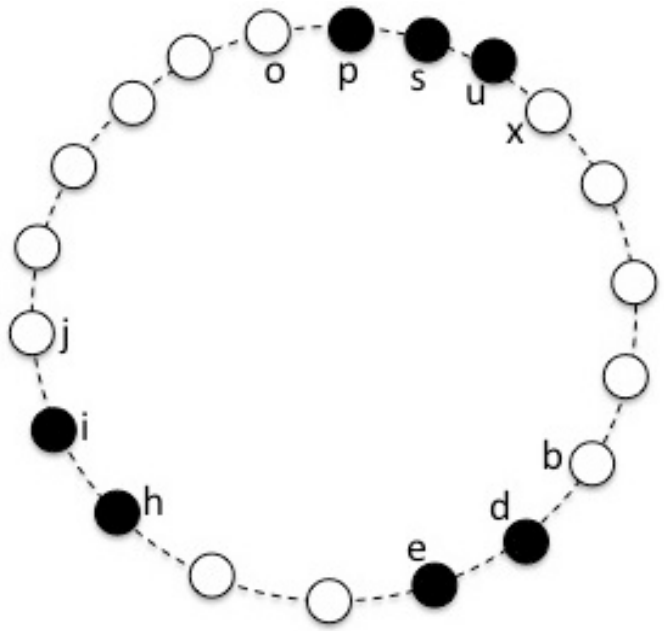
# Third Story
# Network Partitioning without Churn
# (Brief Network Partitions)

# Can our Maintenance Strategies help with Network Partitioning?

| Scenario | Local Maintenance (Correction-on-*, Periodic Stabilization) | Global Maintenance (Merger with passive list/knowledge base) |
|---|---|---|
| **Execution during Network Partition (partition-tolerance)** | Can Create separate rings in each partition but can get stuck. | Merger with KB is Required to provide the best partition-tolerance; however Merger with passive list can fail to fulfill the requirement. |
| **Execution at Partition Repair (merging of overlays)** | Combined reactive and proactive corrections is able to merge multiple overlays, even reacts quicker. | Provides no improvement over the combined local strategies. |

# Execution during Network Partition

- Correction-on-*: unavailability of key ranges.
- Periodic Stabilization: eventually recovers from unavailability, but multiple overlays are formed in same partition.



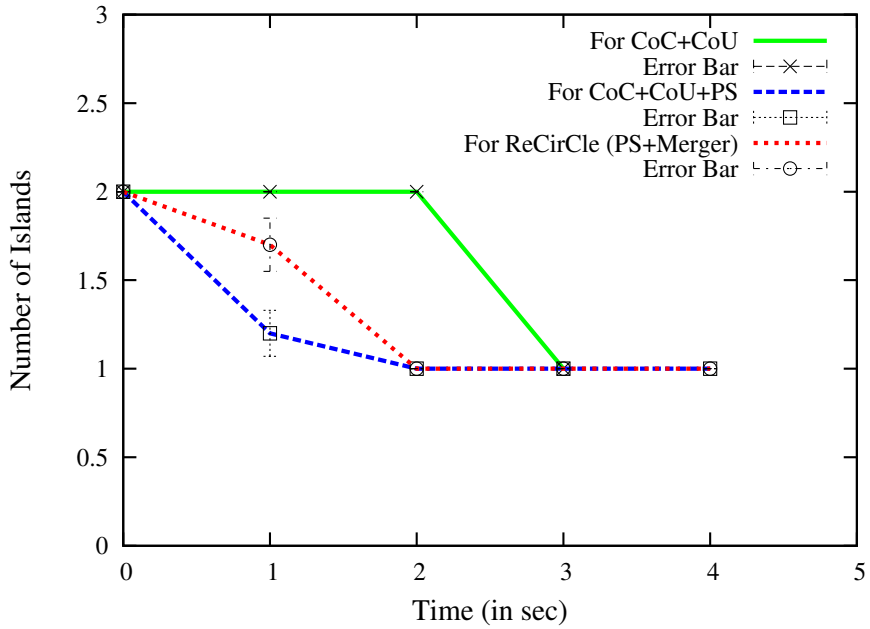

A Partition Scenario: white and black nodes belong to two different partitions; partition having black nodes have absence of more than |succ_list| − 1 consecutive peers (here, |succ_list| = 4).

Merger with passive list fails to trigger merging of multiple overlays in same partition (holding black nodes)

Proactive manner to trigger merging using knowledge base is required.

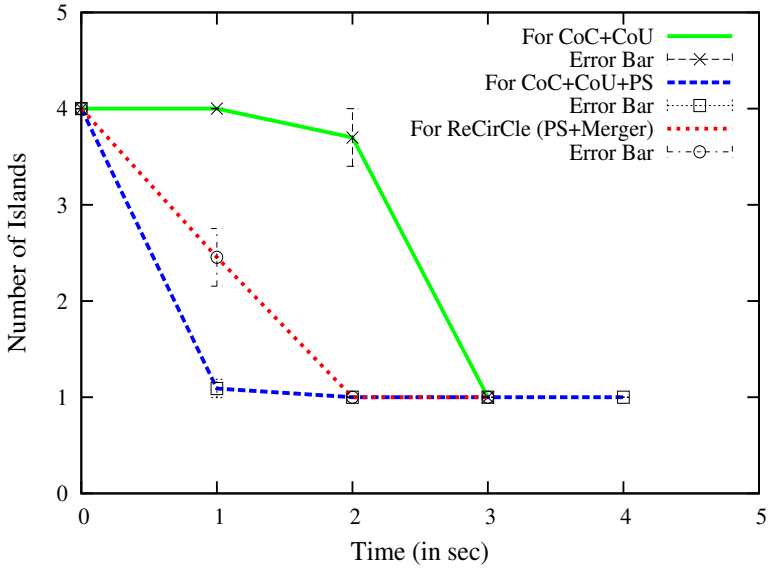
# Execution at Partition Repair (1)

- Evaluation using number of islands as a function of time.
  - An island: a disconnected sub-graph by following successor pointer of each node.

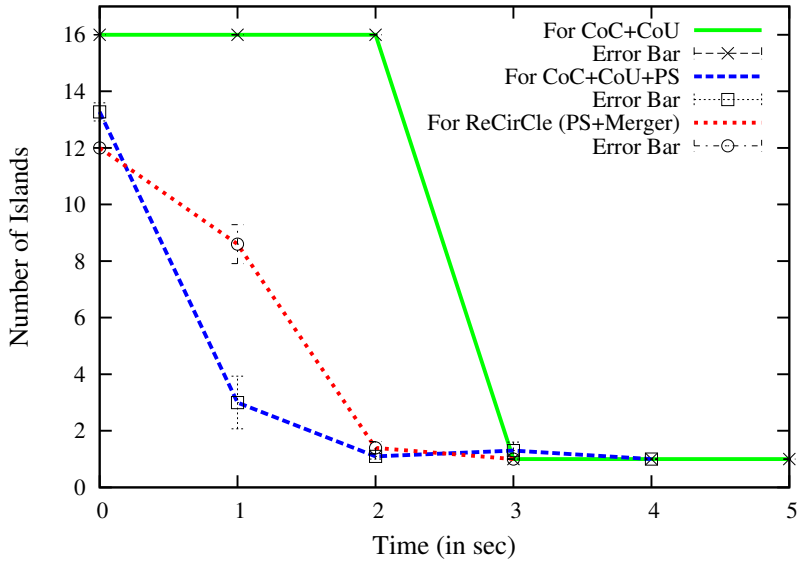



Number of Partitions=2

# Execution at Partition Repair (2)




Number of Partitions=4




Number of Partitions=10

# Fourth Story
# Interaction between Network Partition and Churn

# Stranger Model

- To understand impact of churn during a network partition.
  - Quantify challenge for maintenance mechanism, while merging multiple overlays, as a network partition ceases.

- Due to churn during network partition, partitions of system diverges with time.
  - Suppose $P_1$ and $P_2$ two partitions. $|P_1|=n_1, |P_2|=n_2$ and $n_1 \approx n_2$.
  - Best possible state at t=0, since network partition:

$$(\bigcup_{p_i \in P_1} KB_{p_i}) \bigcap P_2 = P_2$$

# Stranger Model (cont..)

- Assumptions:
  - Uniform distribution of peer lifetimes (corresponds to worst case scenarios).
  - The set, $\bigcup_{p_i \in P_1} KB_{p_i}$, remains unchanged with time.

- After t (t>0) time units, since network partition:

$$(\bigcup_{p_i \in P_1} KB_{p_i}) \bigcap P_2 = n_2 \times e^{-\frac{Ct}{100}}$$

# Cut-off Point

- After $T_{CO}$ time unit $P_1$ and $P_2$ will be complete strangers to each other.

$$n_2 \times e^{-\frac{CT_{CO}}{100}} = 1 \therefore T_{CO} = \frac{100 \times \ln n_2}{C}$$

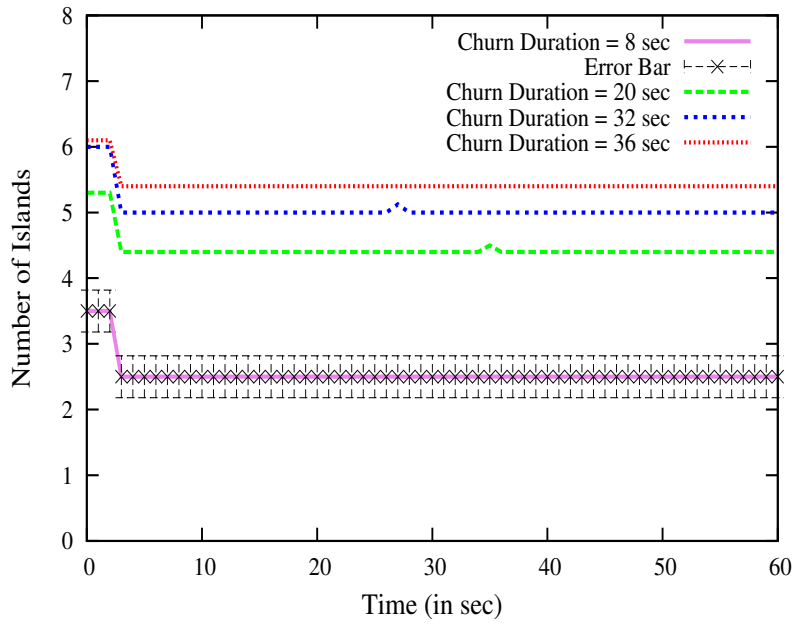
➢ It is impossible for system to achieve reversibility by itself beyond cut-off point.

| Churn | Theoretical $T_{CO}$ | Measured $T_{CO}$ |
|---|---|---|
| 10% | 39.12 | 40 |
| 30% | 13.04 | 14 |
| 80% | 4.89 | 5 |

Validation of Stranger Model by Experiments

# Evaluation of Maintenance Strategies against *Strangers* (1)

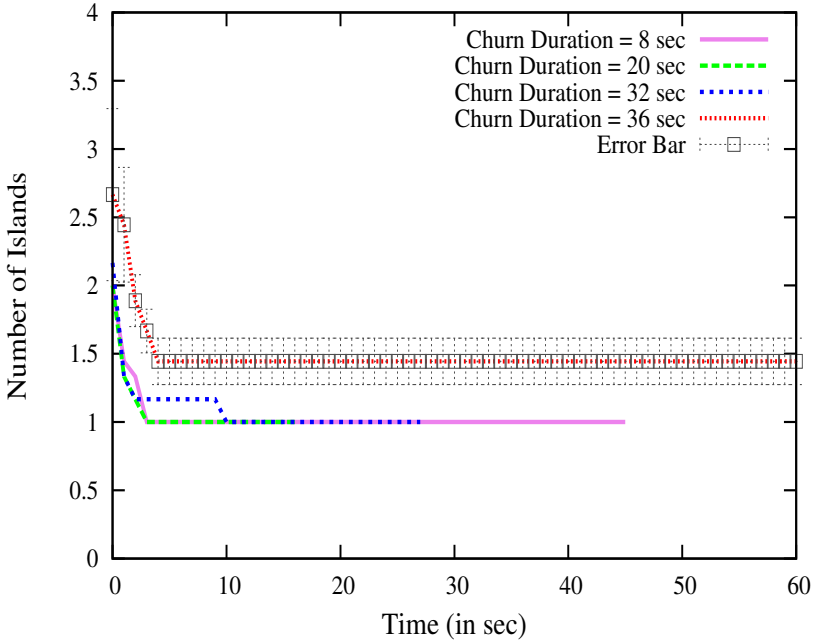
- Operating Environment:
  - Churn = 10%,
  - Partitions= 2,
  - Cut-off Point = 40.
- Metric: Number of Islands (disconnected sub-graph by following successor pointers). Should converge to 1.



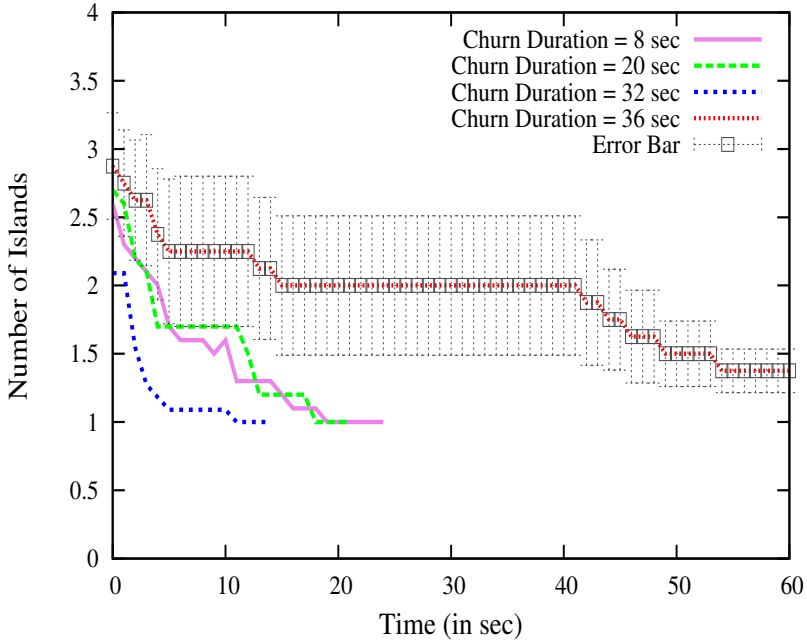

Correction-on-*.
Fails even for lower number of strangers.

# Evaluation of Maintenance Strategies against *Strangers* (2)

32 sec = 77% Strangers, 36 sec= 90% Strangers, 40 sec = cut-off

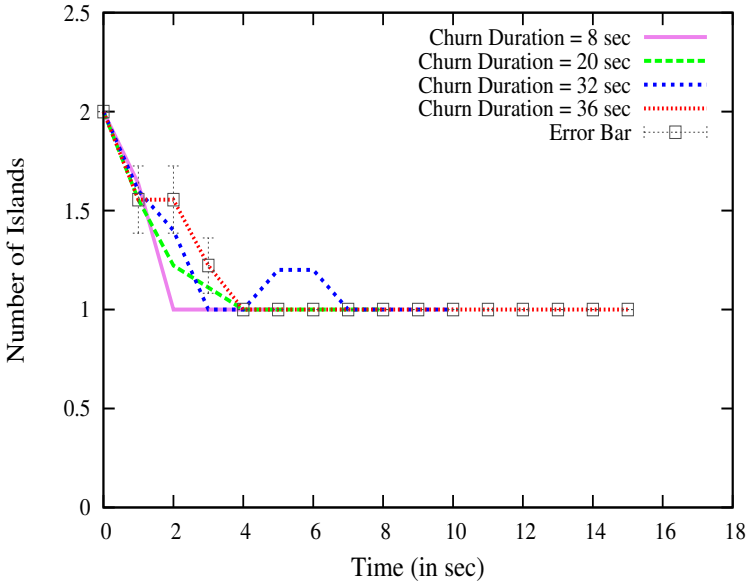



Correction-on-* and Periodic Stabilization. Shows competitive result with an explicit merge algorithm.



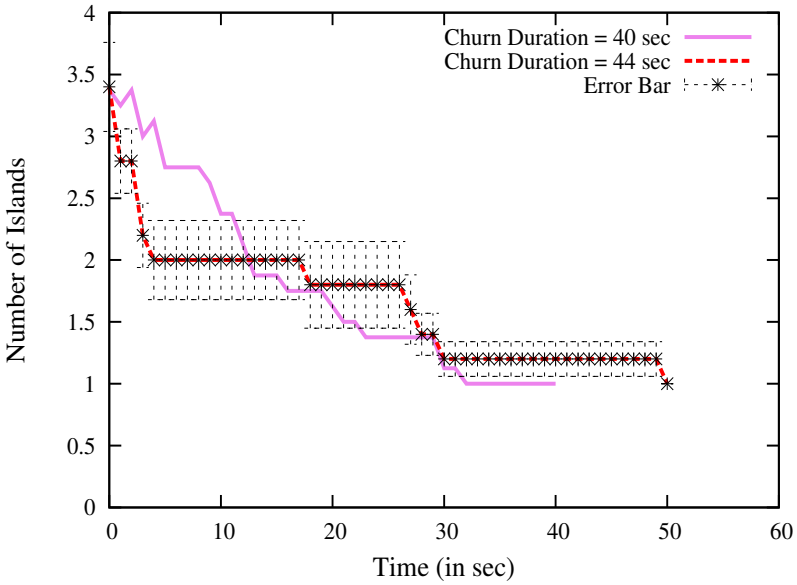

Correction-on-* and Periodic Stabilization and Merger with passive list.
Fails to achieve reversibility for 36 seconds of churn during network partition.

# Getting as close possible to the Cut-off Point : Knowledge Base is the Solution!!




Correction-on-* and Periodic Stabilization and Merger with knowledge base achieve self-healing even for 90% strangers

And Beyond..




Correction-on-* and Periodic Stabilization, Merger with knowledge base and ORACLE to achieve reversibility beyond cut-off point

# Conclusion and Future Work

# Conclusion

- In order to design provably correct complex systems, it is required to study behavior of such systems in inhospitable environments.
  - Build systems that are both predictable (hence, useful in practice) and reversible (hence, they survive).

# Summary of Our Stories!!

- **First Story:** Repeated join and merger using Knowledge Base is required to achieve Reversibility against extremely high Churn.
- **Second Story:** Phase Transitions in the system as a by-product of making the system Reversible (give useful information to applications using APIs).
- **Third Story:** Require Proactive Global Maintenance (Merger with Knowledge Base) during Network Partition. At Partition Repair Local Maintenance is enough to achieve Reversibility as long as the churn intensity remains low or the duration of network partition is short.
- **Fourth Story:** Proactive Global Maintenance (Merger with Knowledge Base) is required to get closer to the cut-off point. Can also inform application when close to cut-off point (by measuring churn and using Stranger Model inside application). With ORACLE it is possible to go beyond the cut-off point (by injecting knowledge using an API).

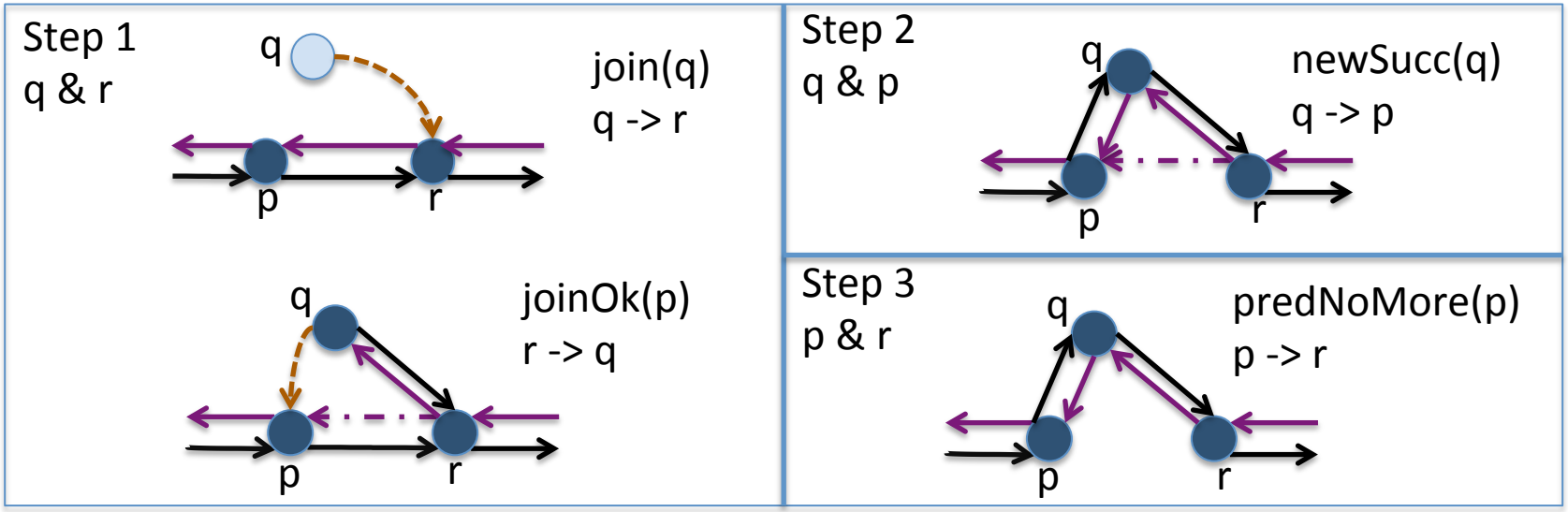
# Future Work (toward a Ph.D.)

- Three main topics:
  - Increase scale (100 nodes is not enough);
  - Increase realism (real system, not simulated);
  - API and application scenarios.
- Maintenance strategies and insights:
  - Need to be validated, extended, increase understanding;
  - Integration with other approaches, e.g., gossip.
- Other stress parameters:
  - Underlying network communication delays: Impact on maintenance and interaction with other stress parameters (churn, network partition);
  - Workload: Impact on maintenance while doing real work.

# Thank You!!

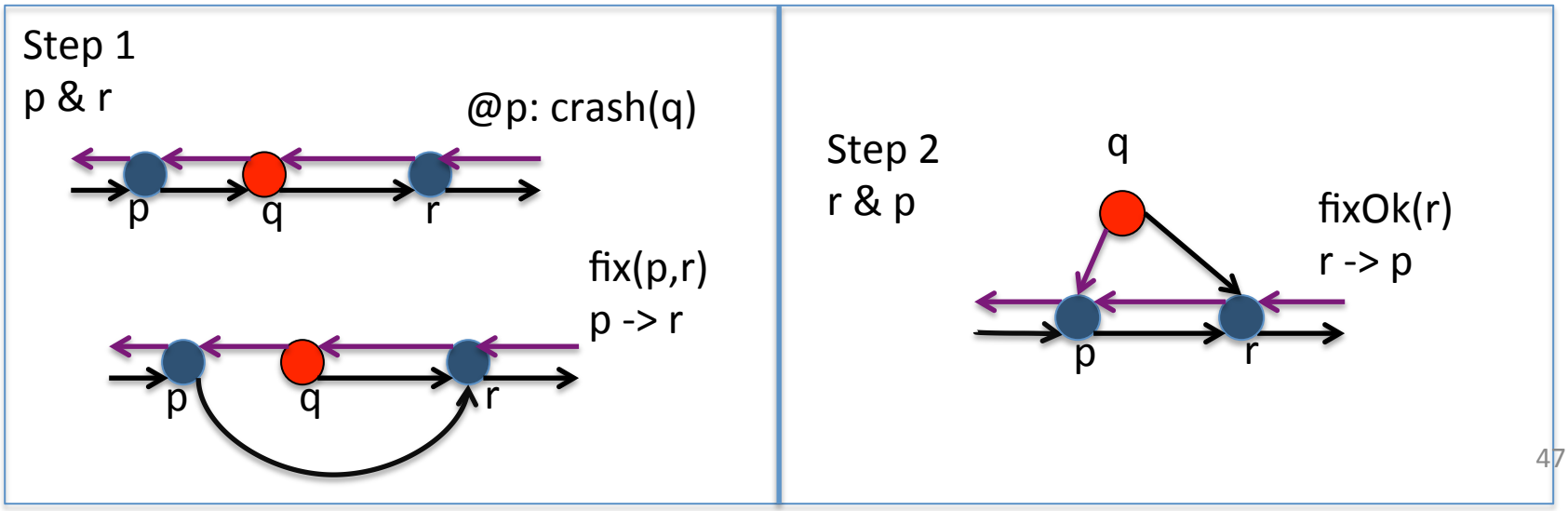
# Questions??

# Correction-on-Change

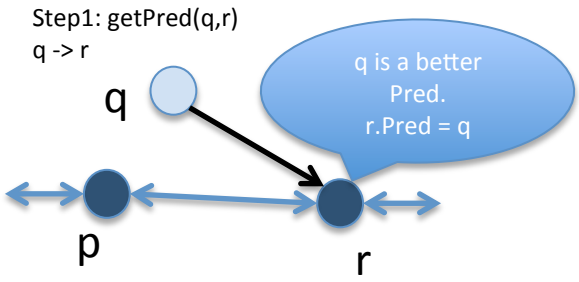



Handling Join

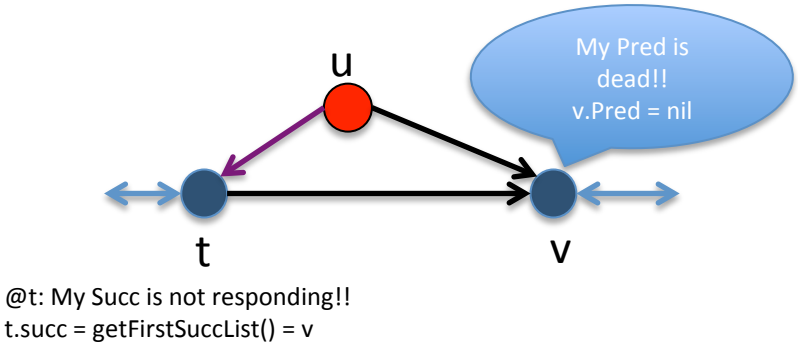
Handling Failure



# Periodic Stabilization




Handling Join



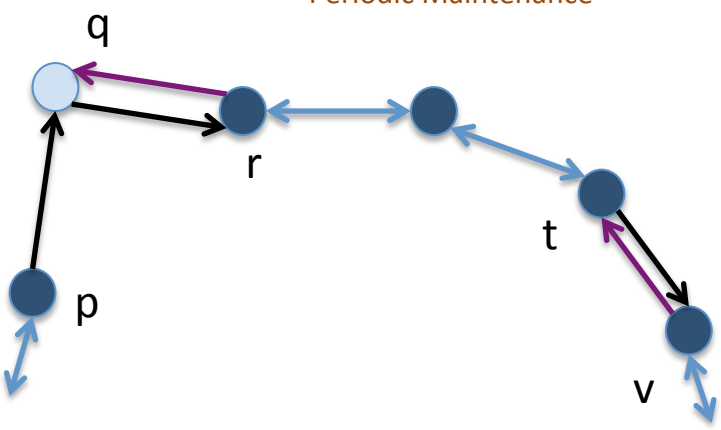

Handling Failure

Step1: getPred(p,r)
p -> r

Step2: getPredRes(q)
r -> p

Step3:
p.succ = q

Periodic Maintenance

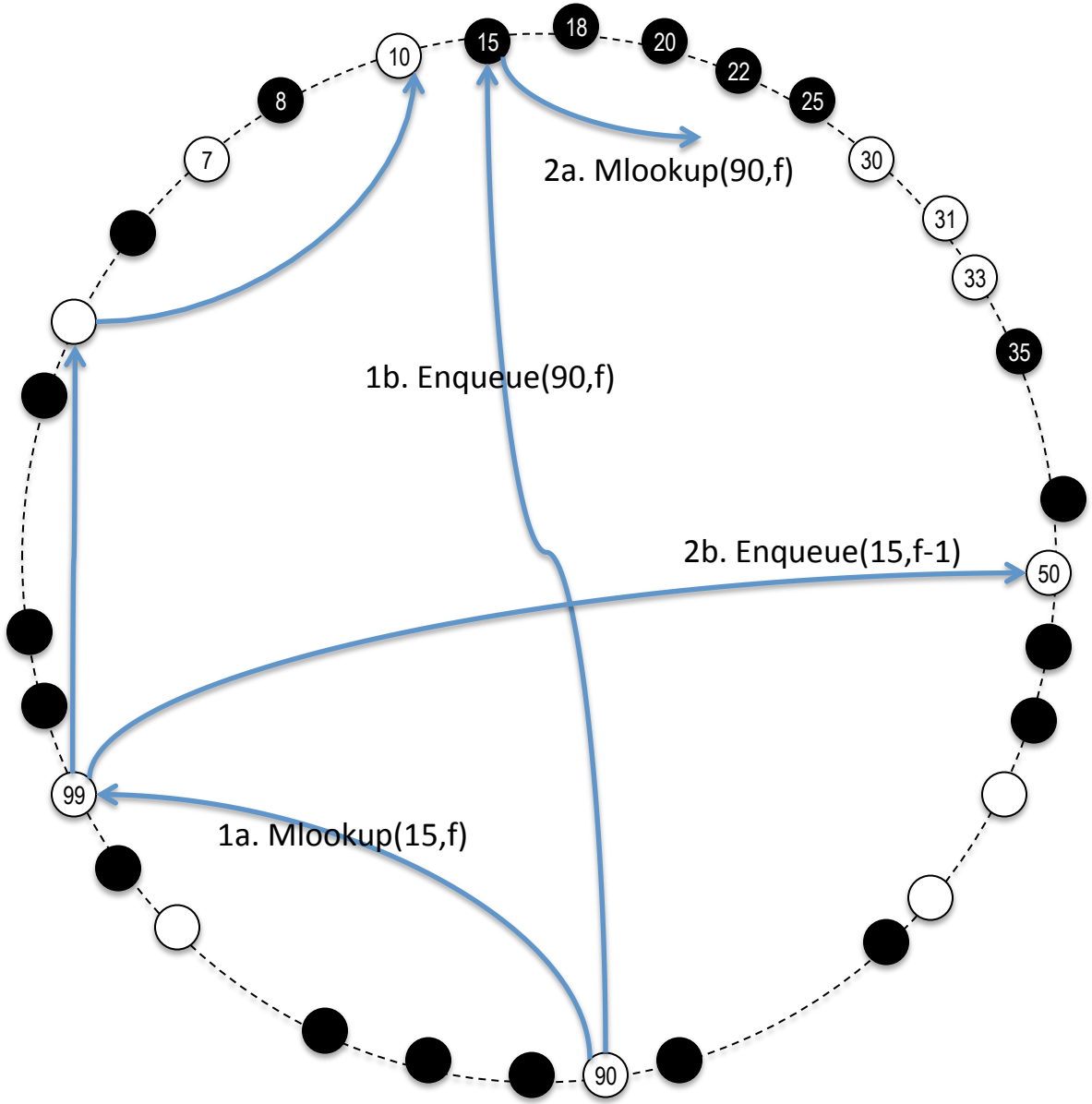


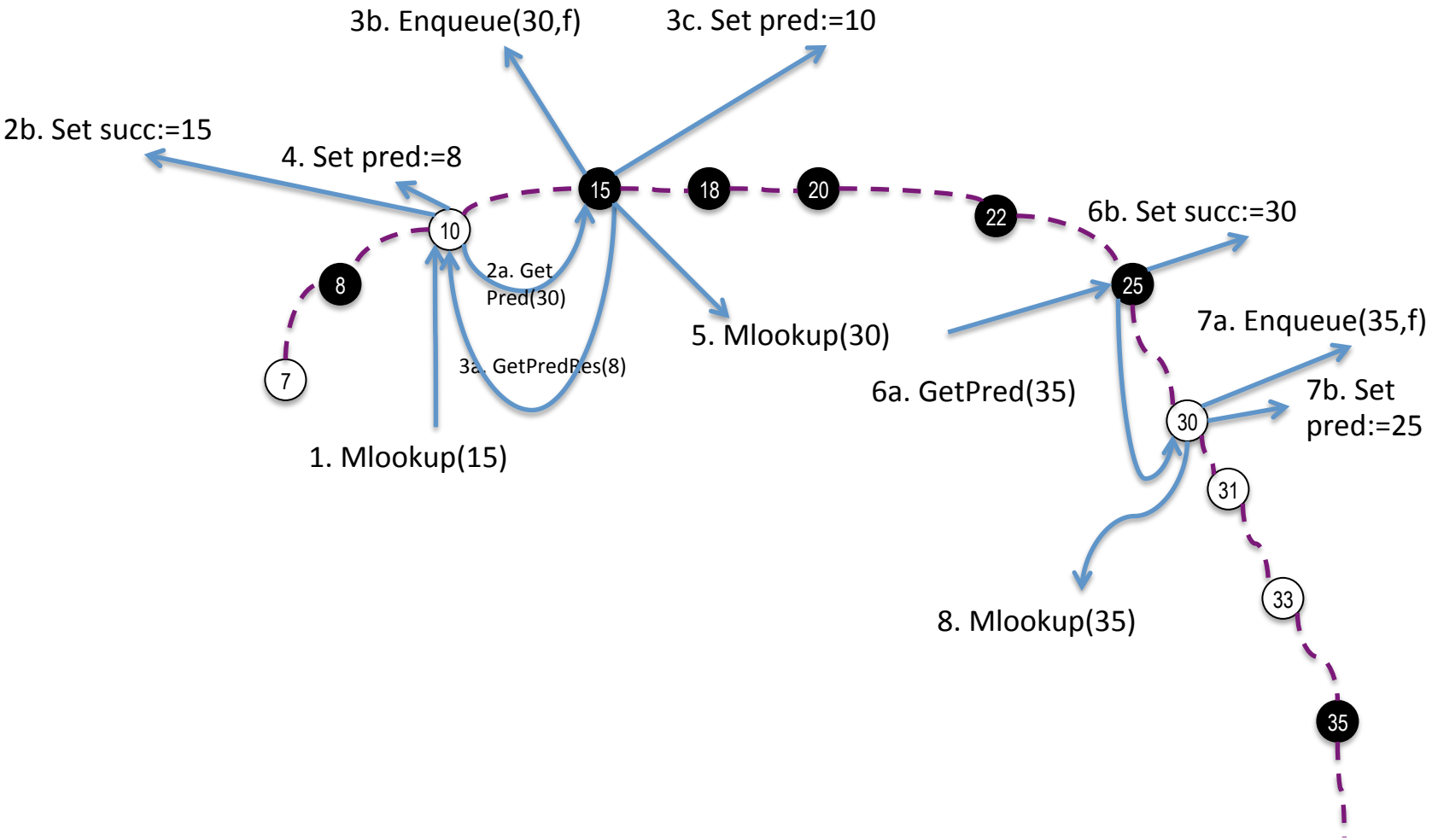

Step1: getPred(t,v)
t -> v

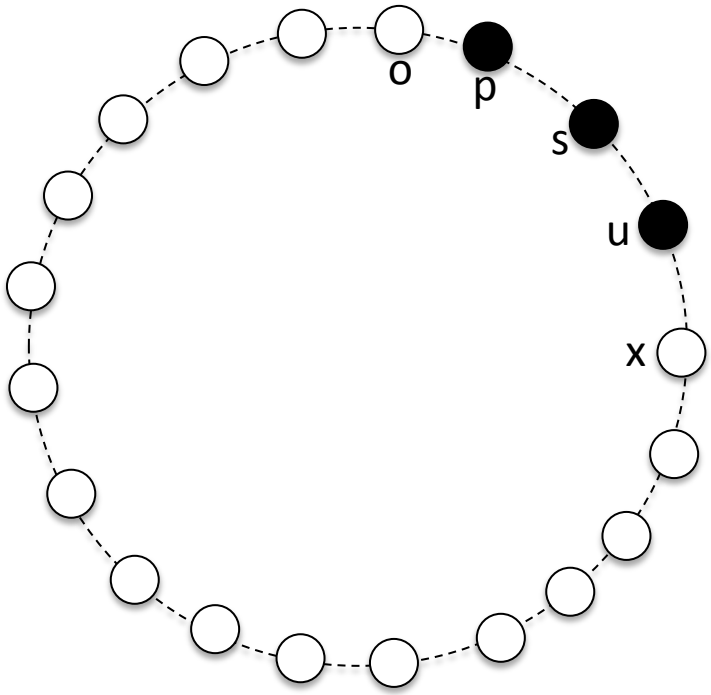
Step2:
v.pred = t
t -> v: getPredRes(nil)

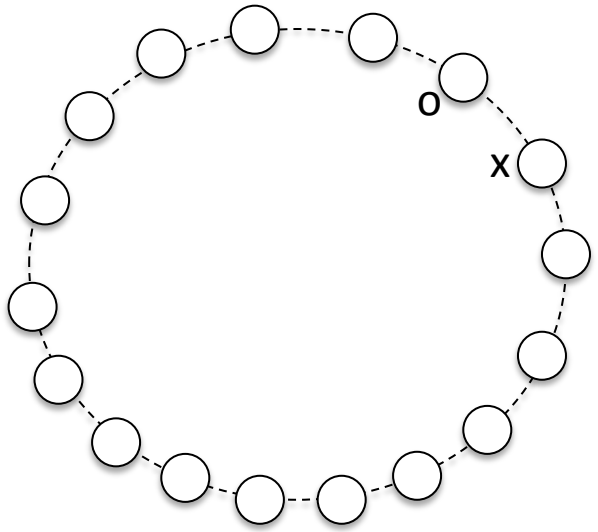
# ReCircle

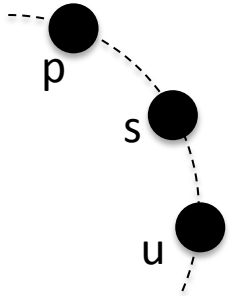
- Extends periodic stabilization to react to extreme events like network partitions and merge.
- Two parts:
  - Periodic Stabilization
  - Merger
- Merger:
  - Each node maintains a queue, which holds identifiers of all nodes that need to be fixed.
  - Every γ time units each node dequeues elements from its queue.
  - Does a greedy lookup using *id* of each element.
  - Upon reaching peer responsible for *id*, triggers same mechanism as periodic stabilization.

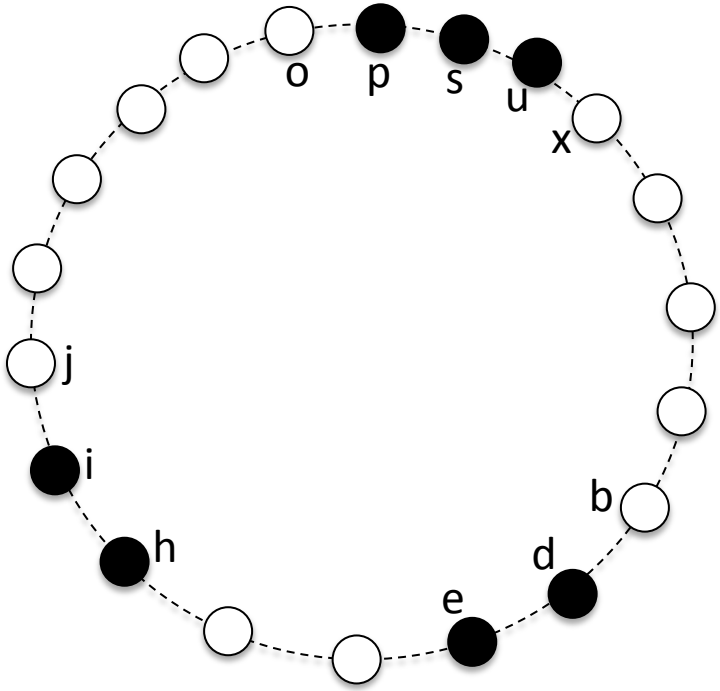

10
15
18
20
22
25
30
31
33
35
8
7
2a. Mlookup(90,f)
1b. Enqueue(90,f)
2b. Enqueue(15,f-1)
50
99
1a. Mlookup(15,f)
90

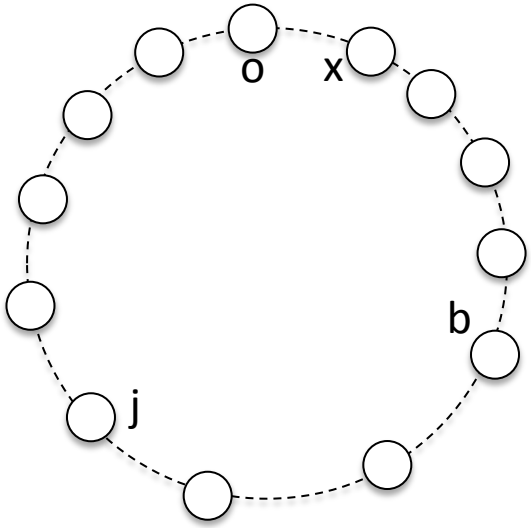

3b. Enqueue(30,f)
3c. Set pred:=10
2b. Set succ:=15
4. Set pred:=8
15
18
20
22
6b. Set succ:=30
10
8
2a. Get Pred(30)
25
7a. Enqueue(35,f)
5. Mlookup(30)
7
3a. GetPredRes(8)
6a. GetPred(35)
7b. Set pred:=25
30
1. Mlookup(15)
31
33
8. Mlookup(35)
35

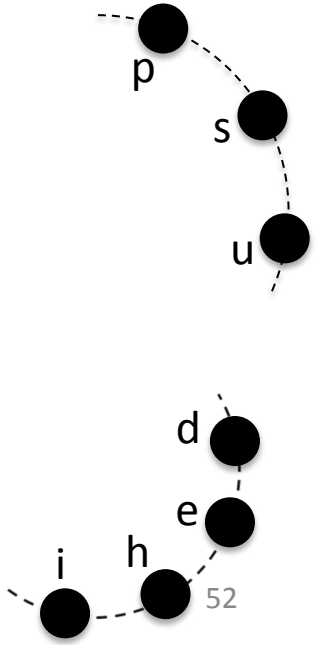

o
p
s
u
x


o
x


p
s
u


o
p
s
u
x
j
i
h
e
d
b


o
x
b
j


p
s
u
d
e
h
i

# Representative Complex Systems

- A particular class of overlays.
  - Logarithmic-style Ring Overlays as per reference architecture proposed by Aberer et. al.[1]
  - Ex. Chord, Chord#, DKS, P2PS, Beernet etc.
  - All these overlays possess the **same key design aspects**.

| Design Aspects | Ring Overlays |
|---|---|
| Choice of Identifier Space | A subset of natural numbers of size N, with closeness metric, $d(x, y) = (y - x) \bmod N$ . |
| Mapping to the Identifier Space | A uniform hash function or some random function. |
| Management of the Identifier Space | A peer with virtual identifier p is responsible for the interval (predecessor(p),p]. |
| Graph Embedding | Belongs to "routing-efficient" small-world networks. |
| Routing Strategy | Greedy routing strategy. |
| Maintenance Strategy | Goal of this work: enhancement of maintenance to survive in inhospitable environments. |



[1]Aberer et. al., "The essence of P2P: a reference architecture for overlay networks," in *Proc. P2P*, 2005.

# Maintenance Strategies

- Can be classified into:
  - ➢ Local Maintenance:
    - ✧ Reactive: with sub-categories correction-on-change (for self-healing) and correction-on-use (provides self-optimization and self-configuration).
      - ❖ Ex. DKS, P2PS, Beernet.
    - ✧ Proactive: correction using periodic probing.
      - ❖ Ex. Chord, Chord#.
  - ➢ Global Maintenance:
    - ✧ Overlay merger algorithm.
      - ❖ Adapted ReCircle[2] for our work.
      - » How to trigger merger?
        - Shafaat et al. proposes a Reactive approach using falsely suspected nodes.
        - We propose a Proactive approach, using **Knowledge Base**.
          - ✓ Each node keeps on learning about other nodes, build an "acquaintance" list.
          - ✓ Optimistically samples in a periodic manner from its knowledge base to trigger merger.



. [2]T. M. Shafaat, "Partition tolerance and data consistency in structured overlay networks," Ph.D. dissertation, KTH, Sweden, 2013.