Interaction Between Network Partitioning and Churn in a Self-Healing Structured Overlay Network

Ruma R. Paul
Université catholique de Louvain, Belgium
KTH Royal Institute of Technology, Sweden
Email: ruma.paul@uclouvain.be, rrpaul@kth.se

Peter Van Roy
Université catholique de Louvain
Belgium
Email: peter.vanroy@uclouvain.be

Vladimir Vlassov
KTH Royal Institute of Technology
Sweden
Email:vladv@kth.se

***Abstract*—We investigate the interaction between *Network Partitioning* and *Churn (node turnover)* in *Structured Overlay Networks*. This work is relevant both to systems with peaks of high stress (e.g., partitions, churn) or continuous high stress. It prepares the way for new application venues in mobile and ad hoc networks, which have high node mobility and intermittent connectivity, and undergo frequent changes in network topology. We evaluate existing overlay maintenance strategies, namely Correction-on-Change, Correction-on-Use, Periodic Stabilization, and Ring Merge. We define the *reversibility property* of a system as its ability to repair itself to provide its original *functionality* when the external stress is withdrawn. We propose a new strategy, Knowledge Base, to improve conditions for reversibility in the case of combined network partitioning and churn. By means of simulations, we demonstrate reversibility for overlay networks with high levels of partition and churn and we make general conclusions about the ability of the maintenance strategies to achieve reversibility. We propose a model, namely Stranger Model, to generalize the impact of *simultaneous* network partitioning and churn. We show that this interaction causes partitions to eventually become strangers to each other, which makes full reversibility impossible when this happens. Using this model, we can predict when irreversibility arrives, which we verify via simulation. However, high levels of one only, network partitioning or churn, do not hinder reversibility. In future work we will extend these results to real systems and experiment with applications that take advantage of reversibility.**

***Keywords*-Network Partition and Churn; Structured Overlay Networks; Partition Tolerance; Ring Overlay Merge;**

I. Introduction

Complex systems, in particular large-scale distributed systems which are an excellent example, consist of many interacting parts and their overall behavior cannot be predicted in a straightforward way from the behavior of each part. They have many operating modes depending on the environment in which they work and what they are supposed to do. This has resulted in many problems when such systems are stressed beyond where their behavior is a straightforward extrapolation of the behavior of their parts. In order to design provably correct complex systems, it is required to study the behavior of such systems in inhospitable environments. We define an *inhospitable environment* as one in which certain stress parameters (such as churn, network partitions, communication delay) can potentially reach high values and temporarily increase without bound. The goal is to build systems that are both predictable (hence, useful in practice) and reversible (hence, they survive) in these environments. This is important for three essential reasons. First, for practical system design it is important to explore highly stressful environments, since even systems running in so-called "stable" environments also have peaks of high stress. Second, it can open new venues for application design, such as mobile and ad-hoc networks, for which current fault-tolerance techniques are insufficient. Third, it is important for scientific reasons, to understand what happens in highly inhospitable regimes. For the practical study, we choose a class of *Structured Overlay Networks (SONs)* defined by the reference architecture of [1]. For this class, we make an empirical study of both the behavior and design in an inhospitable environment. To make this problem manageable, we have chosen *Network Partition* and *Churn* as the stress parameters.

SONs have become a popular way of implementing large-scale distributed systems. The most prominent reason is that the fully decentralized architecture of SONs can easily be made self-organizing and self-healing. Such self-management properties are crucial to deal with the inherent complexities introduced by decentralization. Adding structure provides efficient routing, guaranteed reachability and consistent retrieval of information, however, it also introduces the challenge of maintaining the structure. Among all the structures proposed for SONs, the ring topology is competitive with others in terms of routing efficiency and failure resiliency [2]. However, the temporary inconsistency of the topology poses several challenges for correct operation. Many SONs were proposed to gradually improve and circumvent or relax the requirements of a perfect ring for accuracy; Chord [3], Chord# [4], DKS [5], P2PS [6], Beernet [7] to name a few. All these ring overlays strive to achieve self-healing by using different mechanisms; Chord and Chord# use periodic stabilization, whereas P2PS, DKS and Beernet rely on correction-on-change. However, existing literature lacks an effort to assess the self-healing capability of a SON in case of extreme hostility, e.g., *simultaneous*

network partitioning and churn. Can a SON provide partition tolerance for any scenario of network partitioning and heal itself to regain its structure, once the partition ceases? What are the preconditions to achieve reversibility? The term "reversible" implies that the system is able to regain its original functionality as the stress recedes. However, if the stress is constant, e.g., constant churn, then reversibility does not matter. But this is rare, and probably nonexistent in practice.

We briefly define our two stress parameters. *Churn* is the rate of node turnover. It is due to nodes joining, leaving, and failing. *Network partitioning* is the division of the nodes into multiple disjoint sets, where a node can communicate with the nodes of its own set, but is unable to contact the nodes in the other sets. Several reasons may cause the underlying network to partition: link failure, router issues (failure, misconfiguration, overloading), malicious activities (denial of service attacks), software bugs or physically damaged network equipment [8], [9], [10], failure of "choke points" under natural calamities or similar events [8], policy based issues or conflicts. Due to self-* properties, SONs are expected to provide partition tolerance by forming a separate overlay in each partition. Once the partition ceases, the system should reverse back, i.e., merge the multiple overlays into a single overlay.

***Contributions**:* We introduce *Knowledge Base (KB)* as a new maintenance strategy, that is needed to survive inhospitable operating conditions. The KB at each node is constructed to have a best-effort view of the global membership of the system. We show that in order to ensure partition tolerance for any scenario of network partitions, proactive global maintenance is required along with local maintenance. Using a representative scenario, we demonstrate that multiple overlays can be formed in the same partition. To merge these overlays, an overlay merger needs to be triggered proactively using KB. Our investigation shows that, in case of low churn or brief network partition, only local corrections (combination of correction-on-change and periodic stabilization) at each node suffice for macroscopic healing, i.e., merging of multiple overlays as the partition ceases, without any explicit merge algorithm.

We propose a model, namely *Stranger Model*, to generalize the effect of interaction between network partitions and churn. We validate our model through simulation experiments. Also, we predict and verify irreversibility using this model, by means of the *cut-off* point for a particular churn, beyond which the system is unable to achieve reversibility by itself. We verify the self-healing of different principles to achieve reversibility in this range and identify the preconditions to ensure reversibility up to and beyond the cut-off point. So, the overall contributions of our work are as follows:

- Definition of a new maintenance strategy, KB, and demonstration of its necessity in scenarios of low and high churn during a network partition;
- Identification of the preconditions to ensure partition tolerance of SON for any scenario of network partition;
- Comparative analysis of self-healing of different maintenance strategies as a network partition ceases;
- Demonstration that local corrections (without merger) are sufficient to repair partitions in case of low churn or a brief network partition;
- Identification of the limits of simultaneous partitioning and churn, up to which the system can achieve reversibility by itself, using a model of how the partitions diverge with time (the Stranger Model);
- Demonstration of a reversible SON and identification of conditions needed to make a SON reversible while facing network partition for any duration, with or without churn in between.

The remainder of the paper is organized as follows: in Section II we briefly describe our representative complex systems, namely the class of logarithmic-style ring overlays, Section III and Section IV describe maintenance principles of overlays and their survival and healing for network partition in the absence of churn. In Section V, we analyze the scenarios when churn is present along with network partitioning. Section VI discusses some relevant works and we conclude in Section VII.

II. Representative Complex Systems

We have chosen a particular class of complex systems, a class of SONs following the reference architecture of [1], to conduct our study. For our practical experiments, Beernet [7], an overlay with properties typical of this space, is used. We describe key aspects of our chosen class of SONs and Beernet.

A. Design Space and Representative Design Class

As per [1], there are six key design aspects, using which any overlay network can be characterized. In Table I, we provide key aspects of the class of overlays (referred as ring overlays), to which this work can be directly applied. Our contributions can also be generalized for other classes of overlays having several similar key design aspects.

B. Beernet

Beernet [7] is similar to Chord [3]. Both Beernet and Chord support the addition of the maintenance strategies discussed in this paper. The main difference is that Beernet has a correct lock-free join operation, whereas joins in Chord may be incorrect [12] and joins in DKS [5] (a successor to Chord) require simultaneous locking of multiple nodes. Beernet relaxes the ring membership condition. The neighborhood of each peer of Beernet consists of successor, predecessor, fingers in routing table, a successor list and a predecessor list. The successor list looks ahead along the ring and contains maximum $\log_k(N)$ peers. The predecessor

Design Aspects	Ring Overlays
Choice of Identifier Space: virtual identifier space I, having some closeness metric $d : I \times I \rightarrow \mathbb{R}$.	A subset of $\mathbb{N}$, of size N, with $d(x, y) = (y - x) \mod N$.
Mapping to the Identifier Space: $F_P : P \rightarrow I$ associates peers to a unique id from I and $F_R : R \rightarrow I$ associates resources to ids from I.	A uniform hash function or some random function.
Management of the Identifier Space: $M : I \rightarrow 2^P$ associates to $i = F_R(r) \in I$, the set of peers managing r.	A peer with virtual identifier p is responsible for the interval $(predecessor(p), p]$. The responsibility of a peer may dynamically change due to churn.
Graph Embedding: a directed graph, $G = (P, \varepsilon)$, where P =set of peers and ε =set of edges. A neighborhood relationship $N : P \rightarrow 2^P$, for a peer p, $N(p)$ is the set of peers with which p maintains a connection.	Belongs to "routing-efficient" small-world networks [11]. Each peer p perceives I to be partitioned into $\log(N)$ partitions, where each partition is k times bigger than the previous one. The routing table of p contains $\log_k(N)$ connections/fingers to some nodes from each partition.
Routing Strategy: a non-deterministic function $R : P \times I \rightarrow 2^P$, which at peer p, with neighborhood $N(p)$, for a target identifier i selects the (set of) next peers $R(p, i) \in N(p)$ to forward the message.	Greedy routing strategy. For a target identifier i, peer p selects the closest preceding link, $d \in N(p)$ to forward the message. Since there are always k intervals, routing converges in $O(\log_k(N))$ hops.
Maintenance Strategy: required to maintain the structural integrity while peers go offline or network connection fails.	Goal of this work: enhancement of maintenance in ring overlays to survive and achieve reversibility against inhospitable environments having high levels of both network partitioning and churn.

Table I: Representative Design Class: Reference Architecture for Ring-Based Peer-to-Peer Overlay Networks

list contains all the peers that consider the current node as their successor.

Ring with Branches / Relaxed Ring: Beernet has two invariants: *Every peer is in the same ring as its successor* and *A peer does not need to have connection with its predecessor, but it must know its predecessor's key*. The first invariant allows a new peer to be part of the network by connecting only to its successor. The second one determines the responsibility of a peer. These two properties allow relaxation of the ring: when a peer is not yet connected to its predecessor it forms a branch from the core ring. Fig. 1 shows a Beernet network, where red (dark in B/W) nodes are organized into a ring and green (light in B/W) nodes are on branches. Due to branches, the guarantees about proximity offered by Beernet routing correspond to $O(\log_k(n) + b)$, where b is the distance to the farthest peer on the branch.

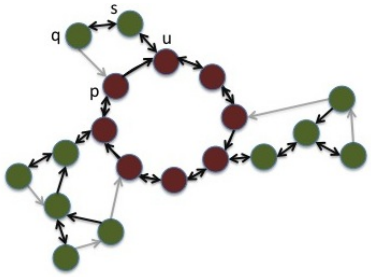


Figure 1: Branches on a relaxed ring. Peers p and s consider u as successor, but u only considers s as predecessor. Peer q has not established a connection with its predecessor p yet.

III. Overlay Maintenance Principles

Following [13], we classify maintenance strategies of overlays into proactive periodic correction (e.g., Chord, Chord#) and reactive mechanisms, e.g., correction-on-change and correction-on-use (as used in DKS, P2PS, Beernet). Also, an overlay merge algorithm is necessary. In our work, we have adapted a partition merger, *ReCircle* [14]. ReCircle has two parts: a periodic stabilization algorithm and a reactive *Merger* that gets triggered in case of extreme events. In addition to these known strategies, we have introduced *Knowledge Base*, which is essential to survive inhospitable environments, like *simultaneous* network partition and churn. Table II summarizes the properties of these principles. We briefly describe each principle.

A. Correction-On-Change & Correction-On-Use

These principles (we refer to them together as *Correction-on-**) were introduced by DKS [5]. Correction-on-change handles join/leave/failure of nodes. Whenever such events are detected, successor and predecessor lists are updated and the correction of pointers (successor, predecessor and fingers) is triggered. Correction-on-use mainly corrects the fingers. Every time messages are routed, information is piggybacked to correct fingers. Thus, correction-on-use provides self-optimization and self-configuration, whereas partial self-healing is achieved through correction-on-change. These principles are fast in terms of reacting and updating local state, which is important to survive an inhospitable environment without introducing any inconsistency. However, without any event, no maintenance is done (i.e., lack of liveness), making them a restricted form of self-healing.

B. Periodic Stabilization (PS)

Chord uses periodic stabilization for ring maintenance, where each peer periodically checks the validity of its predecessor and successor pointers. By exchanging periodic messages with its successor a node attempts to maintain its immediate vicinity. However, this proactive mechanism might be too slow when facing an inhospitable environment. As discussed in [15], [12], lookup inconsistencies and uncorrected false suspicions can be introduced in real implementations. Also, as per [16], there exists a ratio of churn to stabilization frequency, while doing a lookup the longest finger of any peer is always found to be dead, which degrades routing efficiency. To avoid this, it is required to trigger periodic stabilization often, making an inefficient use of bandwidth. Thus, we can say that this proactive mechanism is complementary to reactive correction-on-*: the correction-on-* require no extra messages in case of no failure, but do not work when no event is detected, whereas periodic stabilization needs no event to execute (i.e., having liveness), but is costly in terms of bandwidth consumption.

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 PL	Global	Reactive	Adaptable	Yes	Adaptable
Merger with KB	Global	Proactive	Adaptable	Yes	Adaptable

Table II: Ring Overlay Maintenance Principles

C. ReCircle

ReCircle [14] extends periodic stabilization to react to extreme events like network partitions and merge. It has two parts: periodic stabilization, as described before, and merger. Periodic messages are issued to maintain the local geometry, whereas the merger issues messages that navigate further, triggering awareness and remedying the anomalies, thus ensuring global convergence towards one ring. Once the overlay converges, the messages issued by the merger die out and ReCircle behaves as a normal periodic maintenance algorithm. Here, we provide a brief sketch of the merger.

Each node maintains a queue, which holds the identifiers signifying problematic areas. Every γ time units each node m dequeues the elements from its queue. For each element id, m generates *mlookup(id)*, which is a normal lookup operation. Once it reaches the peer responsible for id, it tries to fix the ring by triggering the same mechanism as periodic stabilization. The fixing is propagated clockwise and also spreaded at random identifiers (selected from the routing table of the current node) on the ring. This is controlled by a knob, called the *fanout* parameter. Thus, the merger can be made adaptable in terms of convergence time. Shafaat *et. al.* in [14] propose "passive list" (*PL*) mechanism to populate the queue at each node. At each node, a list of nodes, called passive list, is maintained that consists of all nodes currently being suspected by that node. Whenever a node is no longer suspected, it is enqueued into the node's queue. Thus, this is a reactive approach to trigger the merger.

D. Knowledge Base

We introduce the new concept of *Knowledge Base (KB)* to trigger the merger of ReCircle in a proactive manner. Triggering the merger only in a reactive manner is not sufficient for reversibility in an inhospitable environment. There might be such partitioning of the overlay that communication problems have not occurred yet and thus no suspicion event has been issued (see Section IV-A). For these scenarios, passive list mechanism [14] fails to trigger the merging. When there is simultaneous network partition and churn, the number of strangers (see Section V-A) increases with time, and merger with passive list fails to achieve reversibility (see Section V-B3). To ensure reversibility up to the best limit for such scenarios and to initiate merging of overlays in the same partition both require more knowledge than a passive list to trigger the merger. The KB at each node is the best-effort view of the global membership of the system. It extends ReCircle [14] in two ways: i) it considers a bigger set of nodes than the passive list, ii) it does proactive triggering of merger, even if there is no false suspicion. The nodes in the KB are not monitored by the failure detector at a peer, so there is no change of the embedded small-world graph of the overlay. There can be different levels of KB:

- Passive KB: Such KB can be built through listening only. At peer p, for each node $a \notin KB_p$ that p comes across (while routing or as a member of its current neighborhood), the virtual identifier and the network reference of a is added to KB_p. The list built by this strategy is never shared with other peers, thus creates no impact on the bandwidth consumption or scalability of the overlay. However, such KB may be out of date quicker.
- Active KB: Using this strategy a node communicates with others to enhance its KB. This can be a weak algorithm, such as each joining node just informing others of its existence, or a stronger algorithm, such as gossip where each node asks a random node to send its KB that it unions with its own KB. The result is that the KB converges faster to maximum information, so that when a partition arrives, it is in the best possible condition for surviving (least possible strangers). However, this approach causes extra load on available bandwidth.
- Oracle: This is information coming from "outside the system". As the duration of the underlying network partition or intensity of churn increases, more nodes in each partition are replaced by new nodes. Thus, eventually the partitions become complete strangers to each other, as $(\bigcup_{p_i \in P_1} KB_{p_i}) \bigcap P_2 = \emptyset$ (and vice versa), here P_1 and P_2 are the sets of peers of two partitions. For such scenarios, KB built at each node falls short to merge the overlays as the partition ceases, which requires the intervention of a third party. An oracle can be part of the API, with which the application layer or a bootstrap server can give information to the system.

To use the KB efficiently, we define a proactive sampling technique that periodically chooses elements of the KB. Since the KB can be very large, it is not practical to use all elements of the KB. Every σ time unit, each node randomly picks up an element e from its KB, where e is neither member of its current neighborhood nor is currently suspected, and enqueues e into its queue. Here, σ is the mean of a Poisson distribution of times when a periodic sampling happens and is a tunable parameter, which affects the convergence time of the overlay merging. This mechanism can be made more efficient by making σ adaptive to the operating environment, which we leave as future work.

Scenario	Local Maintenance (Correction-on-*, PS)	Global Maintenance (Merger with Passive List / KB)
Execution During Network Partition	Can create separate rings in each partition, but also can fail to ensure partition tolerance	Merger with KB is required for partition tolerance; merger with passive list fails to fulfill the requirement
Execution At Partition Repair	Combined local corrections are able to merge multiple overlays, even reacts quicker	Provides no improvement over the combined local strategies

Table III: Experimental results on the limitations of Maintenance Strategies (case of zero churn)

The KB at each node is always growing, so the number of "dead" node references also grows. This introduces a garbage collection problem. We can add a timestamp to all references in KB and optimistically remove all "sufficiently old" nodes from the KB. Implementation and validation of this is left as future work. Also, we intend to improve the maintenance and application of knowledge at each node by borrowing ideas from gossip protocols in our future work.

IV. Network Partition with no Churn

We distinguish two different cases for a SON when there is a network partition: 1) execution during network partition (i.e., partition tolerance) and 2) partition repair (i.e., achieving reversibility). Table III summarizes our results on both cases. Existing works do not analyze case 1 in depth, they only mention the partition scenarios for which a maintenance strategy is able to provide partition tolerance [7]. We have identified the preconditions to ensure partition tolerance for any partition scenario. For case 2, there are several works on ring merge algorithm [14], [17], but no work has been found that assesses partition repair capability of local maintenance principles and compares their performance with an explicit merger. We have analyzed ring merge capabilities of correction-on-* (with and without periodic stabilization) and shown that integration of an explicit merger gains no significant improvement if no churn is induced during partition. In this section, we have only considered zero churn during a partition, which corresponds to a network partition of short duration.

A. Execution During Network Partition

In order to ensure partition tolerance for any partition scenario, proactive global maintenance, i.e., the merger with KB is required along with local ones. We define a SON to be partition tolerant iff i) the peers on each partition form a separate overlay, ii) the system shows high availability, i.e., every lookup must result in a response and iii) each overlay is consistent in itself. Both correction-on-* and periodic stabilization are each able to provide partition tolerance as long as every peer is able to find a valid successor candidate in its successor list, as also identified in existing literature.

We analyze the partition tolerance capability of maintenance strategies when this condition does not hold, using the example scenarios in Fig 2. Correction-on-* is insufficient to provide partition tolerance for these partition scenarios. Unavailability of keys is introduced ($(u, o]$ in Fig 2a and $(u, b]$, $(i, o]$ in Fig 2b) for the partition holding black nodes. Also the nodes fail to form an independent overlay in this partition as per the predefined structure or embedded graph.

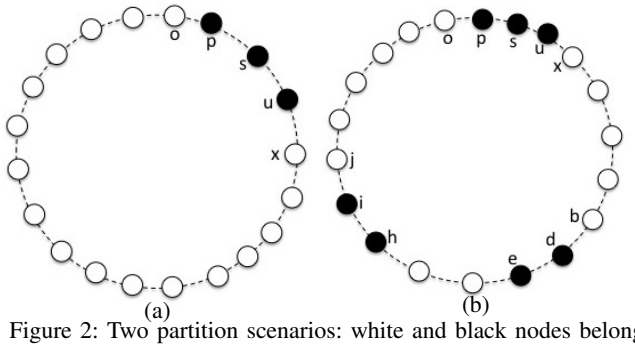


Figure 2: Two partition scenarios: white and black nodes belong to two different partitions. The partition having black nodes, has absence of more than $|succ\_list| - 1$ consecutive peers (here, $|succ\_list| = 4$).

The integration of periodic stabilization eventually heals the gaps in the key range. Thus, for Fig 2a, local maintenance (correction-on-* and PS) is sufficient to provide partition tolerance. However, for Fig 2b two overlays are formed in the partition holding black nodes and remain until the partition ceases, though the nodes on these two overlays are able to communicate with each other. For further explanation refer to [18].

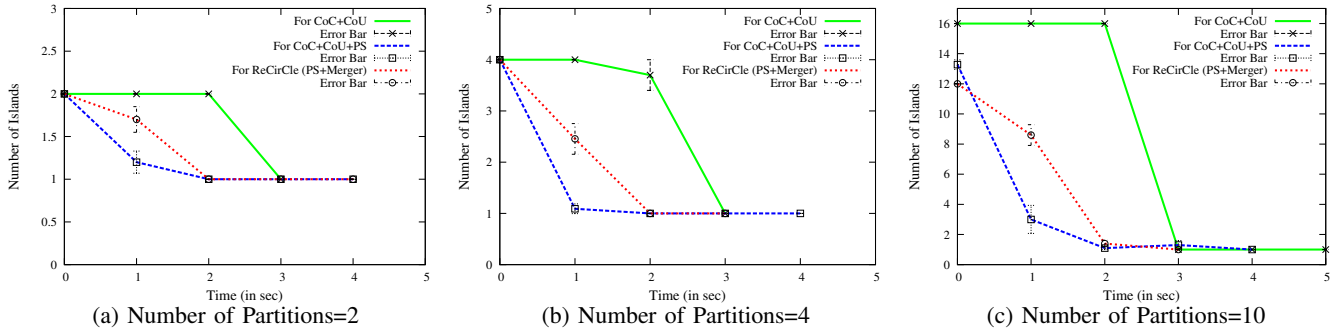
Proactive global maintenance, i.e., the merger with KB, is required to merge multiple overlays formed in the same partition. Using the passive list approach [14] for this purpose fails to trigger the merging process, as nodes in the same partition are not suspected by each other (experimentally verified for the partition scenario in Fig 2b). It is required to trigger the merger in a proactive manner using KB to ensure partition tolerance (also validated via simulation).

B. Execution at Partition Repair

In the case of partition repair, the combination of correction-on-* and periodic stabilization is sufficient to achieve reversibility if there is no churn. This is verified through our experimental result presented in Fig 3. Merger with passive list, as done in [14], does not show improvement over the combined local maintenance strategies.

For our experiments, a SON of 100 peers is used in a simulated environment. The underlying network is simulated by following the empirical distribution of minimum RTT provided in [19]. In order to create P partitions, we create P baskets, where each node of the SON is assigned to a basket with probability $\frac{1}{P}$. During the steady state of the SON, when all nodes are on core ring, we simulate 2, 4 and 10 partitions of the SON, of almost equal sizes. The partition is withdrawn after 30 seconds and we observe the merging of overlays with time.

To quantify the effect of self-healing, we have used metric: Number of *Islands*. We define an island to be a



(a) Number of Partitions=2 (b) Number of Partitions=4 (c) Number of Partitions=10

Figure 3: Number of islands as a function of time (in sec) starting at the moment of partition repair to assess self-healing using different maintenance strategies (case of zero churn)

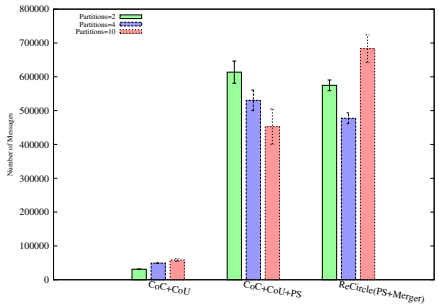


Figure 4: Number of Messages generated for 2, 4 and 10 partitions using different maintenance mechanisms (case of zero churn)

disconnected sub-graph by following the successor pointer of each node. The number of islands can be $\geq$ number of partitions, due to temporary inaccuracy of the successor pointers or inadequate healing of the maintenance strategy. The value of this metric should be 1, for a single run, if overlays are merged together after the partition disappears. However, we continue an experiment until we reach convergence, i.e., perfect ring with no branches. So, the ultimate goal is to make number of islands equal to 1 and all nodes be on core ring. We present the number of islands against time (in seconds) after the partition disappears and an average of 10 independent runs are taken for each second. We remark that the termination time of an experiment, apparent in the result, is the maximum among the samples used. A fixed workload is used in an experiment by injecting transactions, modeled as a Homogeneous Poisson Process with $\lambda = 1$ transaction/sec. A transaction reads one key and updates another one.

Correction-on-* is able to exhibit certain overlay merging capabilities, however, it fails to complete the healing process when the number of partitions goes higher. For experiments with 10 partitions, an average of 91.5% nodes are on the core ring, where rest of the nodes remain on branches. However, this is the least-cost healing mechanism as a response of network merge, as shown in Fig 4.

As we can see in Fig 3, the combination of correction-on-* and periodic stabilization shows the quickest response. We have integrated periodic stabilization with correction-on-* because periodic stabilization by itself is not capable of merging multiple overlays ([3]). The period used in our experiments is 1 second. The impact of the integration of costly periodic stabilization is also clearly visible in Fig 4.

As shown in Fig 3, the integration of a global maintenance, e.g., merger with passive list, does not show any significant improvement over the combined local healing. The period used for dequeuing at each node is 3 seconds and we have kept the *fanout* parameter as 1. However, this is one of the costliest solutions, even with the *fanout* as 1, especially while merging larger number of overlays.

V. Network Partition with Churn

We propose a model to understand the impact of the interaction between network partition and churn. We also predict irreversibility of the system using the proposed model, i.e., the limit (in time unit) beyond which the system is unable to achieve reversibility by itself. Later, we evaluate the reversibility of maintenance principles while facing churn during a network partition and identify the preconditions to achieve reversibility.

In Section IV, we assumed zero churn during a network partition, which is not a realistic scenario for a peer-to-peer network. Though in most existing applications churn remains under a certain limit, as per studies [20], [21], [22], systems with low churn face high peaks and this may happen even during short duration of a network partition. Consider the scenario of a SON running on mobile phones or on an ad hoc network. In such a dynamically changing environment network partition can be a frequent event, along with high churn. However, we have not found any work in literature that demonstrates reversibility for such inhospitable environments, where a SON goes through a network partition, while facing churn at the same time.

A. Stranger Model

We propose a model, namely "Stranger Model", to understand the impact of churn during a network partition. Using this model we quantify the challenge for the maintenance mechanism, while merging multiple overlays, as a network partition ceases.

We start by defining churn: percentage of nodes turnover per time unit (seconds). If we assume equal probability of join and leave events and a single event per time unit, then every other time unit, a node leaves and a new node joins the network, i.e., every other time unit the total number of

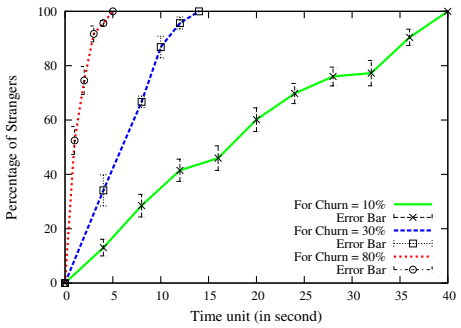


Figure 5: Evaluation of Stranger model for 10%, 30% and 80% of churn

peers is the same, whereas only a single node has a changed identity. So during a network partition every other time unit a new node replaces an existing node in a partition, about which no node of any other partition has any knowledge. This new node becomes a stranger for the nodes on other partition(s). Our model of churn assumes a uniform distribution of peer lifetimes. This is an approximation of a realistic heavy-tailed distribution, where some peers are long-lived. This implies the number of strangers will be less in a real system, which makes it easier for the system to be reversible. Thus, the assumption of uniform peer lifetime distribution is a worst case: our maintenance strategies should work better in practice. We intend to test this in future work.

Suppose the number of nodes on a SON is N. For simplicity we will consider only 2 partitions of almost equal size to present our model. This can be generalized for N partitions. After a network partition, two independent overlays, P_1 and P_2, are formed having n_1 and n_2 nodes respectively on each overlay, i.e., $n_1 + n_2 = N$, $n_1 \approx n_2$ and P_i is the set of nodes of partition i. The best possible starting state for a partition of the overlay is when each partition has complete knowledge about the other, i.e., $(\bigcup_{p_i \in P_1} KB_{p_i}) \bigcap P_2 = P_2$. We present our model to generalize the interaction between network partition and churn, during the partition period, based on this simplifying assumption. For a churn intensity of $C\%$, after 1 time unit, the number of nodes on P_2 known to P_1 is, $(\bigcup_{p_i \in P_1} KB_{p_i}) \bigcap P_2 = n_2 \times \mathrm{e}^{-\frac{C}{100}}$ After t time units, $(\bigcup_{p_i \in P_1} KB_{p_i}) \bigcap P_2 = n_2 \times \mathrm{e}^{-\frac{Ct}{100}}$. This expression assumes the set as $\bigcup_{p_i \in P_1} KB_{p_i}$ is constant. Though this set changes with time, due to the union of KBs of all peers of P_1, the change happens much slower, thus it is safe to assume as constant. Our experimental validation justifies this.

Prediction of Irreversibility: Using the stranger model, we can predict the limit of achieving reversibility, i.e., the number of time units since a network partition beyond which the system is unable to achieve reversibility by itself. For every maintenance principle, merging of overlays is based on the known peers on other partition, with which communication can be established as the partition ceases. So, with the increase of strangers on both P_1 and P_2, the merging becomes more difficult, and at time unit T_{CO}, $(\bigcup_{p_i \in P_1} KB_{p_i}) \bigcap P_2 = \emptyset$ (and vice versa). After T_{CO} time

Churn (C)	Theoretical *Cut-off* Time	Measured *Cut-off* Time
10%	39.12	40
30%	13.04	14
80%	4.89	5

Table IV: *Cut-off* time (T_{CO}) for different values of Churn

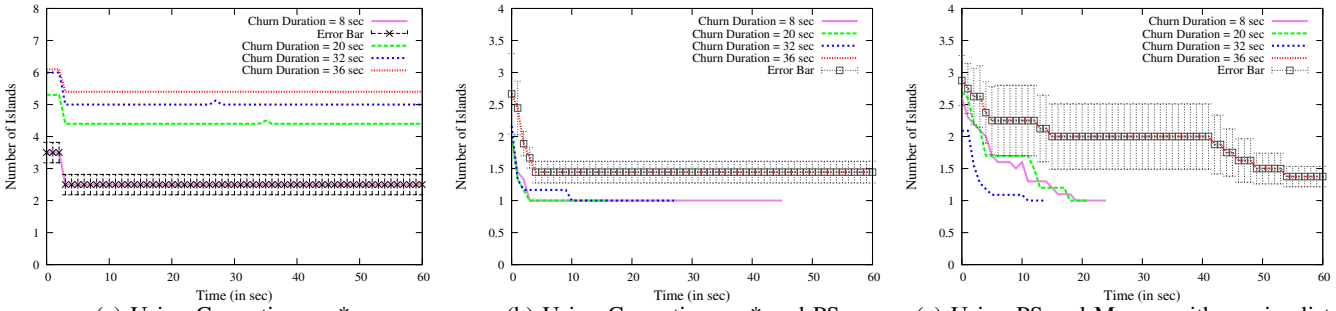
units P_1 and P_2 will be complete strangers to each other. As the nodes on a partition have no reference about any peer on the other partition, no healing mechanism will be effective. We will refer to T_{CO} as the *Cut-Off* point, beyond which third party intervention is required to make the system reversible. We can derive T_{CO} as a function of C and n_2:

$$n_2 \times \mathrm{e}^{-\frac{CT_{CO}}{100}} = 1 \quad \therefore T_{CO} = \frac{100 \times \ln n_2}{C} \qquad (1)$$

Using Equation 1, we can derive T_{CO} for P_2 with respect to P_1 for a given C. We present experimental validation for churn of 10%, 30%, and 80%. For these experiments, a SON of 100 peers is bootstrapped. During steady state, a network partition is simulated by creating 2 partitions of the overlay of equal size, i.e., $|P_1| = |P_2| = 50$. We have verified that just after the partition, i.e., at $t = 0$, $(\bigcup_{p_i \in P_1} KB_{p_i}) \bigcap P_2 = P_2$ and vice versa. Churn of particular intensity is injected for t seconds. To inject a churn event, a partition is chosen with equal probability. We have used correction-on-*, PS and the merger with passive KB for maintenance to have the best measurement of strangers. After withdrawing churn, we retrieve KB of all nodes of P_1 and make their union, S. We compute $|P_2 \setminus S|$ and calculate the % of nodes in P_2 which are stranger to P_1. We do the same for P_1 with respect to P_2 and report the average for a single run. An average of 10 such runs is reported in Fig 5 with increasing time. We present the values of T_{CO} for 10%, 30% and 80% of churn and $n_1 = n_2 = 50$ using Equation 1 and from experiments in Table IV. As shown in Fig 5, the % of strangers increases with time, for all values of churn. Also, the cut-off points are very close to those derived using Equation 1 (see Table IV), thus validating our model. We have expressed T_{CO} assuming the best possible starting state for a partition of the overlay, i.e., each partition has converged knowledge about the other. A generic expression for T_{CO} is subject to future work.

B. Evaluation of Maintenance Principles

We evaluate the maintenance strategies in terms of their abilities to overcome the challenges posed by strangers, while merging multiple overlays. Our results show that Table III remains valid for low percentage of strangers (up to 77%). Beyond that, the merge algorithm with KB is necessary. The effect of churn of different intensities during network partition is the rate of increasing the number of strangers. So, in these experiments, it is sufficient to use only one value of churn, namely 10%, for different durations to create desired percentages of strangers in the system. We use similar experimental setup as described in Section V-A. We inject 10% churn for 8, 20, 32 and 36 seconds, since the



(a) Using Correction-on-* (b) Using Correction-on-* and PS (c) Using PS and Merger with passive list

Figure 6: Number of islands as a function of time (in sec) after withdrawing churn and partition to assess self-healing against strangers

network partition is introduced. Then we withdraw churn, wait for 30 seconds for the healing of the effect of churn on both partitions and restore the network partition. We observe the reversibility of different maintenance mechanisms with time, using the same metric as in Section IV-B: number of islands. We present the average of 10 sample runs in Fig 6; however, we have excluded samples (especially for experiments with 36 seconds) for which the partitions become complete strangers (i.e., $(\bigcup_{p_i \in P_1} KB_{p_i}) \bigcap P_2 = \emptyset$ and vice versa). For stranger measurements, we have constructed a passive KB at each node for these experiments, but the KB is not used for any correction/healing (i.e., to trigger the merger of ReCircle).

1) Correction-on-:* We can see in Fig 6a, correction-on-* principles fail to merge the overlays even with the lowest number of strangers in the system. Also, as the duration of churn increases, i.e., increment of strangers, so is the number of partitions, which remains the same throughout an experiment, after the initial healing during first 3 seconds. This is due to the lack of liveness property of these maintenance mechanisms that self-healing is discontinued.

2) Correction-on- and Periodic Stabilization:* After integration of PS, we can see significant improvement, as apparent in Fig 6b. This combination shows reversibility even for churn duration of 32 seconds; however, beyond that, it fails to guarantee reversibility. For churn duration of 36 seconds, there are runs for which merging has not converged. We have observed till 120 seconds for these runs, but the result remains the same throughout, so for presentation purpose till 60 seconds is shown.

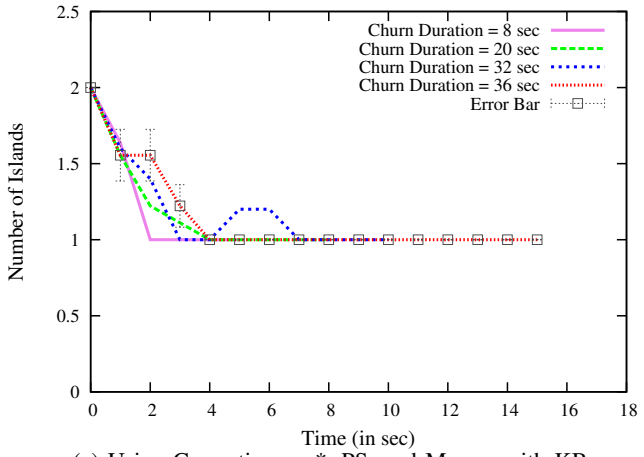
3) ReCircle (Periodic Stabilization and Merger with passive list): As shown in Fig 6c, ReCircle gives no improvement over combined local maintenance (Fig 6b). It shows reversibility up to a churn duration of 32 seconds. However, for 36 seconds of churn duration, there are runs, which have not converged. We have observed till 120 seconds, the result remains the same, so for presentation purpose, till 60 seconds is shown. The reason is the lack of knowledge to trigger the merger using the passive list.

4) Knowledge Base: ReCircle (PS and Merger with passive list) works well up to 32 seconds of churn, i.e., 77% strangers (see Fig 5), whereas adding a passive *Knowledge Base (KB)* works up to 36 seconds of churn, i.e., 90% strangers (and beyond, verification is subject to future work), as shown in Fig 7a. This is much closer to the limit of 100% strangers that is reached at 40 seconds of churn. This clearly shows the effectiveness of the KB. We have used $\sigma = 1$ second to optimistically use elements of the KB to trigger the merger. Also, we have integrated correction-on-* to avoid any inconsistency while handling churn using PS as the only local correction policy. Using correction-on-*, PS and merger with KB gives fast convergence of number of islands to 1. Withdrawing any of these mechanisms will either not converge to 1 or else converge much slower to 1. We have excluded samples for Fig 7a, for which the partitions become complete strangers (i.e., $(\bigcup_{p_i \in P_1} KB_{p_i}) \bigcap P_2 = \emptyset$ and vice versa). Because for these scenarios, this combined healing falls short, as there is no knowledge in a partition about the other to achieve reversibility.

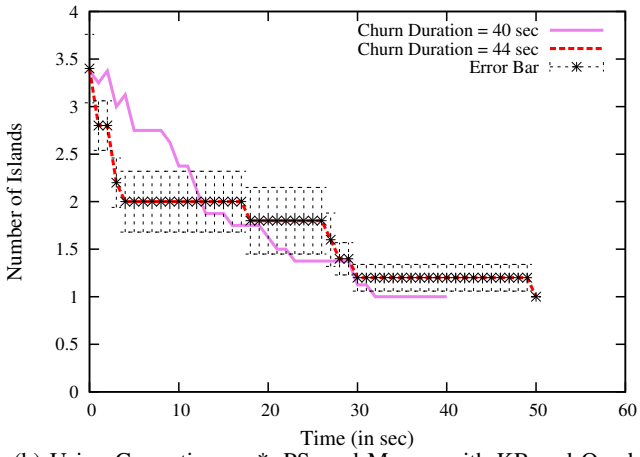
5) Oracle: As we can see in Fig 7b, when the partitions become complete strangers to each other, it is possible to achieve reversibility, as long as an oracle injects the lost knowledge about the other partition. We have implemented an oracle in the application layer, which every 5 seconds, picks up 2 pairs of nodes from its list and introduces them to each other through an API. The list of peers at application layer can be built either an active or a passive way. As per the active approach, the application layer can periodically retrieve KB from the peers it knows, thus build a super-set of these KBs, whereas in passive approach the application layer comes to know about peers while joining. We have used the second approach to build the list at the application layer. Fig 7b shows the convergence for ≥ 40 seconds churn, i.e., 100% strangers (see Fig 5). Also, for these experiments, we have restored network partition immediately, instead of waiting for 30 seconds, in order to observe the overall impact the oracle has on the healing process. For this reason, in the first couple of snapshots the number of islands is higher than 2, as a result of the temporary inaccuracy in the neighborhood of each node, due to churn.

C. Recovery Time and Cost

We present recovery time and cost, in term of number of messages, to achieve reversibility against increasing strangers. We have used a similar experimental setup as



(a) Using Correction-on-*, PS, and Merger with KB



(b) Using Correction-on-*, PS, and Merger with KB and Oracle

Figure 7: Number of islands as a function of time (in sec) after withdrawing churn and partition to assess self-healing against strangers

before and induced churn of 10% for increasing duration during the partition. Correction-on-*, PS and merger with KB (passive KB and oracle) are used as part of maintenance. After withdrawing churn, partition is restored and we measure the time (in seconds) required for complete healing, i.e., all nodes organized into a perfect ring topology. We have used mean value of 20 independent runs for every 4 seconds increase of churn duration. Fig 8 shows the result. This along with Fig 5 gives an idea regarding the percentage of strangers in the system and the time required to achieve reversibility. The recovery time increases with increasing strangers in the system. We report the average number of messages generated in Fig 9. This also follows the similar pattern as in Fig 8. We notice large number of messages generated for a network of 100 peers, raising concern about scalability of the system. However, by controlling the number of messages triggered by the merger using the KB and oracle parameters and setting an optimistic period for PS, the bandwidth consumption can be lowered, while trading-off convergence time.

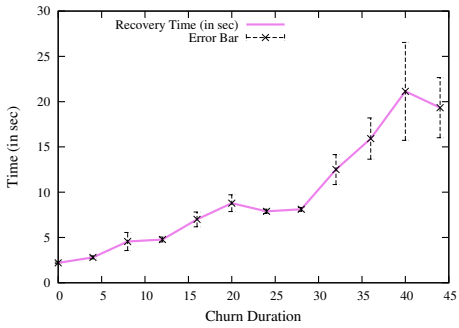


Figure 8: Recovery time for increasing strangers

VI. Related Work

We present some relevant works about network partition and merge in SONs. Several versions of overlay merge algorithm are proposed in Shafaat's Ph.D. thesis [14], of which ReCircle (see Section III-C) is adapted for the purpose of our work. The objective of [14] is to validate the proposed algorithms and sensitivity analysis of different parameters on convergence time and bandwidth consumption. The objective of our work is to assess self-healing achievable using different maintenance strategies and also to identify the limits of the maintenance operations in an inhospitable environment caused by high level of network partition (with or without churn). For this purpose, we analyze the impact of interaction between these two stress parameters. So, the work in [14] and our work complement each other.

In [17], a centralized approach using a bootstrap server is proposed to detect multiple overlays and initiate merge as the underlying network partition ceases. This approach depends on a central bootstrap server; also, it lacks a full algorithm and evaluation of merge process. Merging of multiple P-Grid [23] SONs is presented in [24], [25], which is complementary to our work. Also, there are works on methods/algorithms to merge two independently bootstrapped peer-to-peer overlays [26].

Several works have been done to assess resilience of various maintenance strategies under churn [27], [28], [29]. However, we have not found any work that analyzes the impact of the interaction of network partition and churn, thus to quantify the challenges posed by this interaction on the maintenance strategy of the overlay. In our work, we have presented a model, using which it is possible to generalize the effect of simultaneous network partition and churn. Also, this can provide useful information to the application layer regarding the cut-off point, beyond which the third-party intervention is required to achieve reversibility.

VII. Conclusion

In order to build reliable complex systems, it is imperative to identify required properties of the maintenance strategies of such systems to achieve complete self-healing in the entire operating space. For this purpose, we have chosen one particular class of complex systems, namely a class of *Structured Overlay Networks*, defined by the reference architecture of [1]. We have analyzed the healing capability

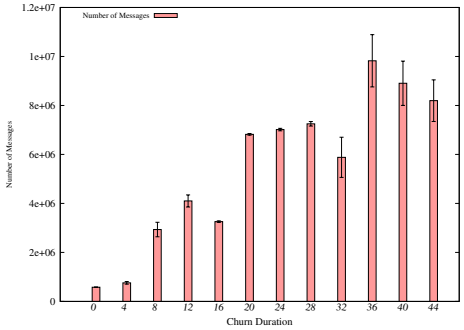


Figure 9: Number of messages generated for increasing strangers

of different maintenance principles in the face of *Network Partition* combined with *Churn*. We have observed that a partition can occur even if there is no communication problem; we also have identified and verified the preconditions to ensure partition tolerance for any scenario of network partition. Our results show that proactive merger using *Knowledge Base (KB)* is needed for execution during a network partition; however it is not needed for partition repair if the percentage of strangers is low; in that case local maintenance is sufficient to achieve reversibility. Reactive merger using Passive List [14] is never better, and it is superseded by its proactive version using KB.

Using the *Stranger Model*, we have generalized the effect of the interaction between network partition and churn. We have identified the boundary (cut-off point) beyond which the system is unable to achieve reversibility by itself. We propose to use a KB to handle partitions under churn (particularly for high percentage of strangers), that contains knowledge collected in a passive way at each node and injected by an oracle beyond the cut-off point.

We keep as future work the efficient self-healing in an inhospitable environment by adapting the system parameters according to the operating environment. Also, validation of our results for large-scale networks is left as future work.

ACKNOWLEDGMENT

This research is partially funded by the *SyncFree* project in the *European Union Seventh Framework Programme* under Grant Agreement no. 609551.

REFERENCES

[1] K. Aberer, L. Onana Alima, A. Ghodsi, S. Girdzijauskas, M. Hauswirth, and S. Haridi, "The essence of P2P: a reference architecture for overlay networks," in *Proc. P2P*, 2005.

[2] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, "The impact of DHT routing geometry on resilience and proximity," in *Proc. ACM SIGCOMM*, 2003.

[3] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proc. ACM SIGCOMM*, 2001.

[4] T. Schütt, F. Schintke, and A. Reinefeld, "A structured overlay for multi-dimensional range queries," in *Euro-Par*, 2007.

[5] L. Onana Alima, S. El-Ansary, P. Brand, and S. Haridi, "DKS (n, k, f): A family of low communication, scalable and fault-tolerant infrastructures for p2p applications," in *Proc. CCGrid*, 2003.

[6] V. Mesaros, B. Carton, and P. Van Roy, "P2PS: Peer-to-peer development platform for Mozart," *Springer LNCS*, vol. 3389, p. 125, 2005.

[7] 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.

[8] F. Jahanian, C. Labovitz, and A. Ahuja, "Experimental study of internet stability and wide-area backbone failures," U. of Michigan, Tech. Rep. CSE-TR-382-98, 1998.

[9] D. Oppenheimer, A. Ganapathi, and D. A. Patterson, "Why do internet services fail, and what can be done about it?" in *Proc. USITS*, 2003.

[10] V. Paxson, "End-to-end routing behavior in the Internet," *IEEE/ACM Transactions on Networking (TON)*, 1997.

[11] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *STOC*, 2000.

[12] A. Ghodsi, "Distributed k-ary system: Algorithms for distributed hash tables," Ph.D. dissertation, KTH, Sweden, 2006.

[13] K. Aberer, A. Datta, and M. Hauswirth, "Route maintenance overheads in DHT overlays," in *Proc. WDAS*, 2004.

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

[15] M. J. Freedman, K. Lakshminarayanan, S. Rhea, and I. Stoica, "Non-transitive connectivity and DHTs," in *WORLDS*, 2005.

[16] S. Krishnamurthy and J. Ardelius, "An analytical framework for the performance evaluation of proximity-aware structured overlays," SICS, Sweden, Tech. Rep., 2008.

[17] G. Kunzmann and A. Binzenhofer, "Autonomically improving the security and robustness of structured P2P overlays," in *Proc. ICSNC*, 2006.

[18] R. R. Paul, "An empirical study of the global behavior of structured overlay networks as complex systems," Licentiate Thesis, KTH, Sweden, 2015.

[19] J. Aikat, J. Kaur, F. D. Smith, and K. Jeffay, "Variability in TCP round-trip times," in *Proc. ACM SIGCOMM IMC*, 2003.

[20] R. Bhagwan, S. Savage, and G. Voelker, "Understanding availability," in *Proc. IPTPS*, 2003.

[21] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proc. MMCN*, 2002.

[22] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *Proc. IMC*, 2006.

[23] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt, "P-Grid: a self-organizing structured p2p system," *SIGMOD Record*, vol. 32, no. 3, pp. 29–33, 2003.

[24] A. Datta and K. Aberer, "The challenges of merging two similar structured overlays: A tale of two networks," in *Proc. IWSOS*, 2006.

[25] A. Datta, "Merging intra-planetary index structures: Decentralized bootstrapping of overlays." in *Proc. SASO*, 2007.

[26] ——, "Merging ring-structured overlay indices: toward network-data transparency," *Computing*, vol. 94, no. 8-10, pp. 783–809, 2012.

[27] D. Liben-Nowell, H. Balakrishnan, and D. Karger, "Analysis of the evolution of peer-to-peer systems," in *Proc. PODC*, 2002.

[28] R. Mahajan, M. Castro, and A. Rowstron, "Controlling the cost of reliability in peer-to-peer overlays," in *Proc. IPTPS*, 2003.

[29] S. Krishnamurthy, S. El-Ansary, E. Aurell, and S. Haridi, "An analytical study of a structured overlay in the presence of dynamic membership," *IEEE/ACM TON*, 2008.