# An Empirical Study of the Global Behavior of A 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*—Distributed applications built on top of *Structured Overlay Networks (SONs)* operate based on certain assurances from the underlying Peer-to-Peer network. Such applications continue to increase in scale, becoming more complex and difficult to manage. The situation becomes worse if the behavior of underlying SON is non-deterministic or even unknown for a given scenario. This implies non-trivial questions: what behavior should the application layer expect from the underlying SON in a given scenario? Under what conditions should a SON exhibit resiliency against an extremely hostile environment? Ideally, the behavior of a complex system such as a SON needs to be defined for every possible operating condition. Existing literature lacks a systematic and in-depth study of the global behavior of a SON. This work is a step towards answering those questions, which starts by proposing an organization of the global operating space of a SON, also defines the term "behavior", with respect to a SON.**

**In order to conduct the experimental study, an existing ring-based SON, namely Beernet, is chosen as a representative. As the entire operating space of a SON is extremely large, due to space limitation, this paper presents the first results of our investigation, the behavior of Beernet along the dimension of *Churn*. The study assesses behaviors like key availability, updates, replica management, and failed transactions, as a function of churn up to $100\%$ node turnover per time unit. The result shows that, continuous injection of extremely high churn causes the ring to be dissolved, creating isolation of peers. However, at such a high node turnover of $100\%$ per $5s$, there are instances, where the $\%$ of failed transactions didn't reach $100\%$, especially in cases, when the join events dominate failures during initial period.**

## I. Introduction

The *Structured Overlay Networks (SONs)* have become the most popular way of implementing large scaled distributed systems because of their de-centralized, fault-tolerance and self-organizing properties. The application layer requires several assurances from the underlying SON. It is required for a large-scale system to be deterministic in all scenarios, especially the adverse ones. For that, it is necessary to have an underlying SON, whose behavior is well known in every situation. However, current literature lacks such effort to explore the behavior of a SON in its entire operating space. The existing works mostly validate the operations of the SON in usual or some specific (like simultaneous failures, interleaved join and failures) scenarios. With the increased development of large-scaled distributed applications, the behavior of a SON needs to be assessed in every possible operating environment so that the critical points in the operating space can be identified, beyond which the resiliency of the SON starts degrading.

In order to explore the entire operating space of a SON, the first step is to identify and organize it, which has been addressed by this work. Among all the structures proposed for SONs, the ring topology is the most popular choice; as it is competitive with other SONs in terms of reaching any other node in smaller steps and also most resilient to failures [1]. Many ring-based SONs were proposed; Chord [2], DKS [3], Beernet [4] to name a few. We have chosen Beernet as a representative ring-based SON for this study because it is a typical exponential-routing SON, so the outcome of this study can be applied to any other SON with similar design. The goal is to define the best behavior of a SON at any point of its operating space, in other terms, to identify the pre-conditions of a SONs resiliency against an extremely hostile environment. The next key term in our title is "behavior", so we have identified a set of behavior matrices to provide an idea of the affect of a particular environment on the operation of the SON. So, the overall contributions are as follows:

- Organization of the entire operating space of a SON;
- Present a self-adaptable eventually perfect failure detector; evaluate its *Quality-of-Service (QoS)*;
- Augment Beernet's replica management with a protocol for better availability and consistency of replica sets;
- Define behavior of a SON in terms of a set of parameters and present the behavior of Beernet along Churn.

The rest of the paper is organized as follows: Section II presents the organization of a SON's operating space, Section III gives a brief description of Beernet. In Section IV, we propose all the matrices. Section V presents the experimental results. Section VI discusses some relevant works and we conclude in Section VII.

## II. Global Operating Space

As the operating space of a SON is large, it is essential to organize it. Most of the space is constituted by the scenarios when something goes wrong, so we can call this organization as the fault model for a SON. We attempt to structure all possible non-malicious failure conditions using $5$ dimensions. Any point in this $5$-dimensional space represents a valid operating environment that a SON may face during its lifetime. Below we present these dimensions and only describe in detail the parameter along churn:

978-1-4799-6201-3/14/$31.00 

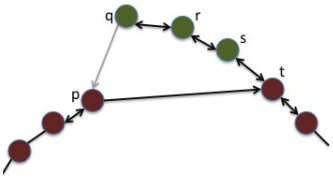


Fig. 1: A branch on the relaxed ring as peer q cannot communicate with p. Peers p and s consider t as successor, but t only considers s as predecessor.

- Churn: most usual and basic scenario that a SON faces during its lifetime.
- Network Partition: creates distinguishable operating condition for a SON.
- Network Dynamicity: triggers false suspicions, in the face of which SON struggles to maintain its structure.
- Workload: beyond a threshold might make a particular SON unresponsive.
- Ring Size: evaluates the scalability of SON.

### A. Churn

The term *churn* is used to express the measurement of peers joining or leaving the network. The literature mostly addresses churn as the only failure scenario. In most cases churn is introduced under a certain limit. However, studies [5], [6] show that P2P environment experience high churn rates. So, it is worth observing the behavior of a SON under high churn.

Let us define churn, assuming that join and leave events have equal probabilities. We say that a node changes its identity under churn with two events, join and leave. Assuming also a single event per time unit, every 2 time units, one node leaves and a new node joins the network; thus, only a single node changes its identity whereas the total number of nodes in the network does not change. That's why, we measure churn as the percentage of nodes turnover per time unit.

## III. Beernet

Here we describe the concepts behind Beernet (for more details, see [4]) and present the extensions and modifications: a self-adapting QoS-aware eventually perfect failure detector and a lazy data-migration protocol during failure recovery. The relaxed ring structure of Beernet doesn't rely on transitive connectivity, or perfect failure detection. It uses *Correction-on-change* and *Correction-on-use* [3] to obtain self-configuration, self-optimization and self-healing. Every step of Beernet's join and leave operations requires agreement between only two peers. Lookup consistency is guaranteed after every step. Beernet doesn't rely on graceful leaving of peers.

Beernet has 2 invariants: *Every peer is in the same ring as its successor*, that allows a new peer to be part of the network by connecting only to its successor and *A peer does not need to have connection with its predecessor, but it must know its predecessor's key*, this determines the responsibility of the peer. These properties allow relaxation of the ring in the form of branches as shown in Fig. 1.

The routing principle is a slight variation of Chord's: *a peer always forwards the lookup request to the responsible candidate*. Because of branches, the proximity guarantee of the Beernet lookup mechanism corresponds to $O(log_k(n)+b)$, where b is the distance to the farthest peer on the branch.

### A. Failure Detector

*Failure Detectors* play an important role in network management, because changes like node crashes or leaves are detected by failure detectors. Beernet relies on *Eventually Perfect Failure Detection*. Maintaining *Quality of Service (QoS)* of failure detectors is crucial for Beernet's performance, because frequent false suspicions may cause instability in SON; and longer period to detect a crash or correct a false suspicion causes larger inconsistency. This section presents a self-adaptable eventually perfect failure detector, which adapts to the environment, providing QoS.

The public release of Beernet (version 0.9) [7] includes a generic eventually perfect failure detector [8]. The algorithm has 2 observed problems. First, the timeout is global for all connections of a node whereas Round Trip Time (RTT) is often different for each connection; thus, the detection process is driven by the slowest connection. Second, the timeout only increases, whereas RTT may vary along time, in a long running distributed system. It is necessary to adapt timeout to RTT, without creating any oscillation.

The expectations from the failure detector are threefold: to detect failures quickly, to reduce the number of false suspicions, and to correct mistakes quickly. These goals are contradictory: the time to detect failures or correct a mistake depends on the timeout period. Lower timeout delays lead to quicker detection of failures or correction of mistakes, but in order to reduce the number of false suspicions, timeout needs to be large enough to cope with changes in the environment. To determine the best value of timeout period for each connection, adapting to RTT of each connection is the only option.

Based on above reasoning, we propose a self-adapting failure detection algorithm, influenced by [9], that adapts to RTT of a connection and keeps a safety period to accommodate the variability or unpredictability of a large and complex network like Internet. A weighted standard deviation over the history of RTT is the best candidate as the safety period, where the weight $m$ depends on the variability in the network. The failure detector at each node maintains a history of last $k$ RTT for each connection and the timeout period of next round for each connection is determined based on this history as follows.

Timeout Period for next round = Average RTT over last $k$ rounds + $m$*Standard Deviation of RTT over last $k$ rounds.

Our algorithm addresses the issues mentioned above by striving to attain an optimal timeout. The algorithm can be expressed as a function of $k$ and $m$, the best values for these parameters can be determined for a particular environment, based on experiments, as shown in Section V. For the initial rounds, when there is not enough RTT data collected, a predefined timeout period is used.

### B. Transactional DHT

Beernet provides support for Transactional Distributed Hash Table (DHT) with replication. This section provides relevant information about Beernet's data layer support with an augmentation of replica management in order to achieve improved

data availability and replica consistency. Paxos Consensus protocol is used for atomic commits in the transactions.

Beernet uses Symmetric Replication [10]. The replica management of Beernet strives to maintain a consistent set of replicas under any extreme environment. Let us analyze the operations of this layer under churn. When a new peer joins, the successor pushes all data-items to the new peer that it has become responsible for. Taking values only from a single replica is fine in this case, as if the successor has stale value of a data-item, this will replace one bad replica with one bad replica (i.e. the number of bad replicas stays the same). However, when there is a failure, it is more important to read from the majority during the recovery, as there is no way for the recovery node to know whether the dead peer was up-to-date or not. Instead of doing expensive transaction for each data-item we propose an inexpensive lazy-migration protocol during failure recovery, which can achieve eventually consistent replica set for a data-item, without creating conflict with simultaneous transaction of that particular data-item.

*1) Lazy Data Migration:* Each data-item has a version number, which increases on each update. The *read* operation reads from the majority and returns the value with the highest version number. While committing a transaction, a replica votes for commit if $version_{new} >= version_{existing}$ for an existing data-item or the data-item is absent. The monotonicity of the version number, along with the advantages of symmetric replication is exploited in the lazy migration protocol. Suppose peer $q$ has successor $r$ and predecessor $p$. When $p$ suspects $q$, it initiates failure recovery, if $q$ is also suspected by $r$, $r$ takes over the responsibility for all data items that $q$ was responsible for. With the symmetric replication, $r$ can find out other members of the replica set of the corresponding data items, eliminating the need of any expensive group management of replica sets. Though there is a replica set per data-item, due to symmetric replication, many replica sets overlap, that facilitate the data migration. A pull request to do data migration for the corresponding data-items is sent by $r$ to all other members of the replica set. If a peer receives a pull request, it retrieves all the data-items belonging to the specified range and sends them to the destination. After receiving data, $r$ does an update if $version_{received} > version_{existing}$ for an existing unlocked data-item or the data-item is absent in $r$. Thus, $r$ may have a stale value for the data-item temporarily, however eventually $r$ will be consistent for the data-item. Though, this is not a perfect solution to achieve consistent and complete replica set for each data-item, as there might be non-overlapping replica sets, however this is a trade-off between cost and consistency.

## IV. Matrices

We study the "behavior" of a SON in all possible non-malicious scenarios. In Section II, we have defined the operating space for a SON, here we define the term "behavior", with respect to a SON. We propose a set of matrices, which capture the behavior of a SON at various levels. We also define the parameters, which can be used to assess the QoS of an eventually perfect failure detector, discussed in Section III-A.

### A. Behavioral Matrices

For a systematic and thorough understanding of the behavior of a SON with transactional DHT, we have identified 3 levels. Below we present the parameters of all levels, studying which will provide an assessment of the impact of a particular environment on the operation of a SON:

- Data Level: % of failed transactions, % of lost keys, % of inconsistent replicas, % of lost updates.
- Connection Level: # of imperfections introduced, % of time a node experiences imperfections, % of nodes on core ring.
- Routing Level: # of messages generated.

The data level parameters signify the impact of a particular environment on the DHT and data level operations. The last parameter, % of lost updates is the ratio of successful updates of the stored data in a particular environment. This parameter is mostly dominated by % of failed transaction, however other scenarios may also contribute. If the replicas holding up-to-date value of a data-item leave or crash or become unavailable (for example, due to partition) simultaneously then there will be loss of updates, though in this case there were no failed transactions. Another scenario: suppose transaction is designed for an application in such way that the update operation of a key depends on its old value, in this case if the key is lost, then it doesn't invoke any update, contributing to the lost updates. All other data level parameters are apparent from their titles.

The connection level parameters assess the deviation of the ideal ring structure during adjustment with a particular environment. The parameter # of imperfections introduced, counts the total number of times peers falsely suspect their ideal successor or predecessor and choose an imperfect peer. The next parameter % of time a node experiences imperfections accounts for how long a peer goes through such imperfection. The parameter % of nodes on core ring reflects the rigidity of the ring by finding out the maximal ring in the system and reports % of nodes on it. The routing level parameter shows how many messages are generated to cope with the changed environment. These matrices are general enough to be applied for any SON with transactional DHT.

### B. QoS Matrices for Eventually Perfect Failure Detection

In order to assess QoS of an eventually perfect failure detector we define 3 primary QoS matrices, which are in-line with those proposed by Chen et al. [11]:

- Detection Time: Average duration after which all peers permanently suspect the crashed node.
- Accuracy: % of suspicions that are correct.
- Reaction Time/Mistake Duration: Average duration after which a peer detects and corrects a false suspicion.

## V. Experimental Studies

We present the evaluation of Beernet only along *Churn*. The study for other dimensions (Section II) is ongoing work. Before doing any experiment, it is necessary to evaluate and tune the failure detector (Section III-A).

We have ported Beernet $v0.9$ [7] implemented in Mozart-Oz ($v1.4$), to Mozart 2.0 [12]. In our experiments we have used the simulated environment in order to have an in-depth understanding of a SON along a particular dimension of the operating space; we leave the experimental study in a real-world dynamic environment to future work. To simulate underlying network, the fixed end-to-end delays are set based on the empirical distribution of minimum RTT provided in [13]. A network of 100 peers is used in all presented experiments.

### A. Self-Adapting Failure Detector

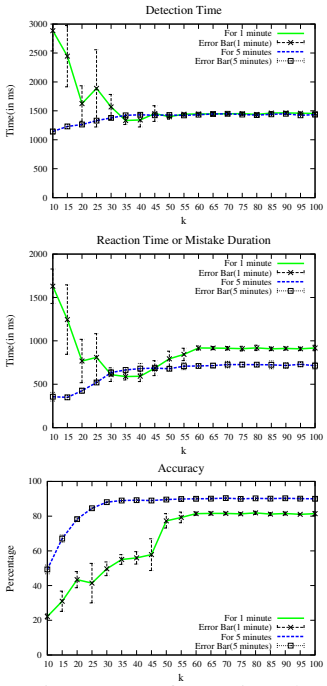


Fig. 2: QoS for various k

To evaluate the QoS of failure detection, in the steady state of SON, 5% churn is injected every $5s$. The churn events are modeled by a *Homogeneous Poisson Process (HPP)* with $\lambda = 2$ event/sec (i.e. for 100 peers, 5% churn issues 10 events per $5s$). Simultaneously, to simulate variability in the underlying network, 5% of links (end-to-end overlay connections) changes connectivity every $5s$, based on the empirical distribution of standard deviation in per-connection RTTs as measured in [13]. The connectivity change events are also modeled as an HPP with $\lambda = 5 * L/100$ per 5s, where $L$ is the total number of connections (for 100 peers, $L = \binom{100}{2}$). Also, to evaluate the adapting capability of the failure detector, we have taken measurement for 1 minute and 5 minutes.

For this environment, we have measured the QoS parameters by varying RTT history size, $k$ (with $m = 4$) and the weight of the standard deviation of the RTT history, $m$ (with $k = 30$). The obtained results are shown in Fig. 2 and Fig. 3. For the following experiments, we have kept $k = 30$ and $m = 4$.

### B. Behavior of Beernet along Churn

Churn of increasing intensity is injected in order to understand Beernet's resiliency in extremely hostile

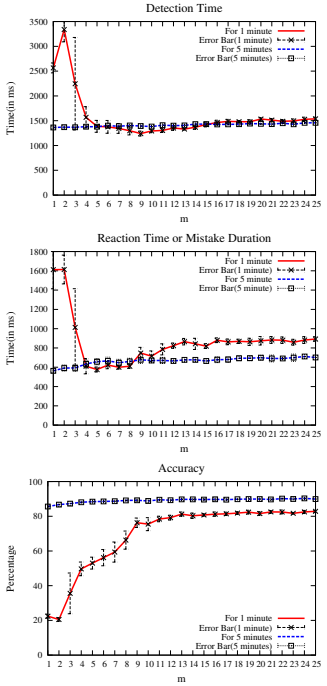


Fig. 3: QoS for various m

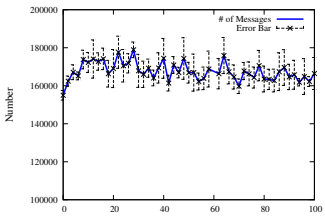


Fig. 4: Number of messages

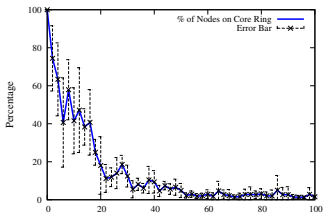


Fig. 5: Rigidity of SON

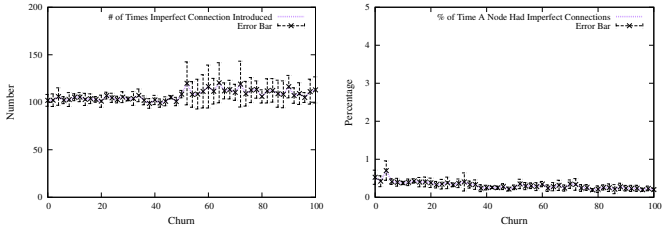


Fig. 6: Imperfections in SON

environment, where nodes are joining and leaving abruptly in a continuous and frequent manner. The result presented is the outcome of Beernet's operation using Correction-on-change and Correction-on-use. The impact of Lazy-Data-Migration (Section III-B1) on the data level parameters is also evaluated.

A workload is a stream of transactions, modeled as an HPP with $\lambda = 1$ transaction/sec. A transaction reads one key and updates another one. The replication factor is 4. To assess the impact of churn only, the workload is kept in such way that without churn there are no failed transactions and inconsistencies. The churn is modeled as described in V-A.

*1) Routing Level Parameter:* Fig. 4 shows the impact on the number of messages generated for increasing churn.

*2) Connection Level Parameters:* Two parameters: the number of times imperfections introduced and percentage of time a node experiences imperfections, reflect the impact of variability in the underlying network, so increasing churn should not have any impact on these, as depicted by Fig. 6. However, percentage of nodes on the core ring shows high sensitiveness against churn, as shown in Fig. 5. With the increase of churn the rigidity of the ring degrades, because the inter-arrival time between two churn events becomes too small to complete any recovery. The system shows state changes that can be compared with the state of matter in Physics: *Solid*, *Liquid* and *Gas*. When there is no churn, 100% nodes are on the core-ring, which corresponds to the Solid state, the ring is rigid. As churn increases, there are nodes on branches, can be compared with Liquid state of matter, which is still valid state for Beernet. When the environment becomes extremely hostile, the ring start dissolving, where the nodes become isolated, corresponding to the Gas state.

*3) Data Level Parameters:* To evaluate the impact of the Lazy-Data-Migration protocol all data level parameters are measured with-without lazy-migration for each churn value. Fig. 7-10 show 4 data-level parameters for the 2 result sets.

The degradation of ring topology leads to more failures of data layer operations, as found in Fig. 7. As expected, lazy data migration does not create any conflicts with transactions. The main impact of lazy migration is clearly visible in Fig. 9 and Fig. 10; the integration of this low-cost, optimistic approach

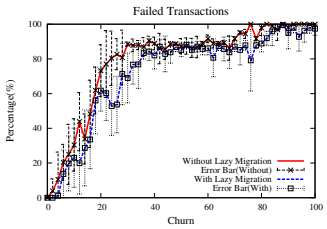


Fig. 7: % of failed transactions

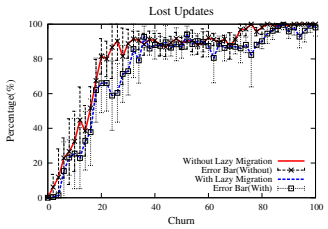


Fig. 8: % of lost updates

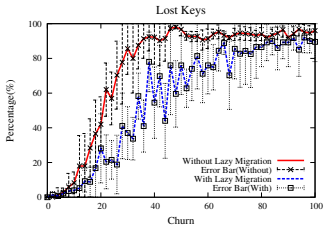


Fig. 9: % of lost keys

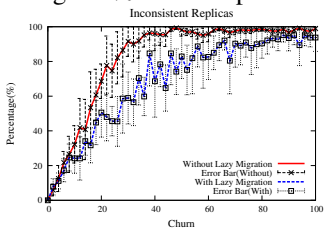


Fig. 10: % of inconsistent replicas

leads to less key loss and inconsistent replicas. Fig. 8 follows the trends of Fig. 7, as explained before, more churn leads to more failed transactions i.e. more updates are lost.

## VI. Related Work

This section briefly describes some most relevant works about failure detection and SONs. Among 2 implementation strategies for timeout based failure detectors [14], [15], the ping strategy provides finer-grained control, also matches with the design philosophy of Beernet. Existing works [11], [14] on QoS of a failure detector are mostly based on heartbeat strategy, compared to which the ping strategy has twice more variables that negatively affects the quality of timeout period estimation [14]. Very few works exist on the QoS of ping based failure detectors. A message efficient algorithm is provided in [16], but each detector monitors only a single process. In [17], [18] the status of the monitored process can be known by making a query to the failure detector, whereas Beernet needs failure detector to push notifications regarding any event.

The most relevant work on the performance evaluation or dependability analysis of SON is found in [19], that evaluates routing level consistency and improved performance of MSPastry (a new implementation of Pastry [20]) by varying parameters like network topology, node session times, link loss rates, and amount of application traffic. Another work on lookup consistency, [21] evaluates the frequency of inconsistent lookups, overlapping responsibilities and unavailability of keys in Chord [2] resulting from unreliable failure detectors and churn. All these works are about the routing level parameters and validate the correctness and improvement of lookup consistency; however a systematic and multi-layer evaluation of a SON is of utmost importance. There are also analytical works, like [22], [23]. Our experimental study can be used as a complementary or precondition of such work.

## VII. Conclusion

Due to the increasing complexity of distributed applications running on top of *Structured Overlay Networks (SONs)*, it has become imperative to define the behavior of the underlying SON at each point of the operating space. However, existing literature lacks such comprehensive or systematic study. This work proposes an organization of entire operating space of a SON using 5 dimensions and defines the "behavior" of a SON in terms of 3-level matrices. In order to achieve the best possible behavior of a SON, an existing ring-based SON, Beernet is extended and modified. As part of which, a QoS-aware self-adapting failure detection algorithm is presented and evaluated. Also, a lazy data migration protocol is included as part of failure recovery to fill up the gap in Beernet's replica management. Finally, the evaluation of Beernet along *Churn* using the defined behavioral matrices is presented.

## References


[1] 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.

[2] 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.

[3] L. O. 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.

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

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

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

[7] Programming Languages and Distributed Computing Research Group, "Beernet: pbeer-to-pbeer network," http://beernet.info.ucl.ac.be, 2009.

[8] C. Cachin, R. Guerraoui, and L. Rodrigues, *Introduction to Reliable and Secure Distributed Programming*, 2nd ed. Springer, 2011.

[9] "RFC 6298: Computing TCP's retransmission timer," http://tools.ietf.org/html/rfc6298.

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

[11] V. Chen, S. Toueg, and M. K. Aguilera, "On the quality of service of failure detectors," *IEEE Transactions on Computers*, 2002.

[12] "The Mozart-oz programming system," http://www.mozart-oz.org, 2013.

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

[14] M. Bertier, O. Marin, and P. Sens, "Implementation and performance evaluation of an adaptable failure detector," in *Proc. IEEE DSN*, 2002.

[15] B. Satzger, A. Pietzowski, W. Trumler, and T. Ungerer, "A new adaptive accrual failure detector for dependable distributed systems," in *Proc. ACM SAC*, 2007.

[16] M. Larrea, S. Arévalo, and A. Fernández, "Efficient algorithms to implement unreliable failure detectors in partially synchronous systems," in *Proc. DISC*, 1999.

[17] C. Fetzer, M. Raynal, and F. Tronel, "An adaptive failure detection protocol," in *Proc. IEEE PRDC*, 2001.

[18] C. Fetzer, U. Schmid, and M. Süsskraut, "On the possibility of consensus in asynchronous systems with finite average response times," in *Proc. ICDCS*, 2005.

[19] M. Castro, M. Costa, and A. Rowstron, "Performance and dependability of structured peer-to-peer overlays," in *Proc. IEEE DSN*, 2004.

[20] A. Rowstron and P. Druschel, "Pastry:scalable, distributed object location and routing for large-scale peer-to-peer systems," in *Proc. IFIP/ACM Middleware*, 2001.

[21] T. M. Shafaat, M. Moser, T. Schütt, A. Reinefeld, A. Ghodsi, and S. Haridi, "Key-based consistency and availability in structured overlay networks," in *Proc. InfoScale*, 2008.

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

[23] 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 Transactions on Networking*, 2008.