A Topology-Aware Approach for Distributed Data Reconciliation in P2P Networks

Manal El Dick, Vidal Martins, Esther Pacitti

A Topology-Aware Approach for Distributed Data Reconciliation in P2P Networks

Manal El Dick[1], Vidal Martins[1,2], and Esther Pacitti[1]

[1] ATLAS Group, INRIA and LINA, University of Nantes, France
[2] PPGIA/PUCPR - Pontifical Catholic University of Paraná, Brazil
manal.el-dick@univ-nantes.fr, firstname.lastname@univ-nantes.fr

Abstract. A growing number of collaborative applications are being built on top of Peer-to-Peer (P2P) networks which provide scalability and support dynamic behavior. However, the distributed algorithms used by these applications typically introduce multiple communications and interactions between nodes. This is because P2P networks are constructed independently of the underlying topology, which may cause high latencies and communication overheads. In this paper, we propose a topology-aware approach that exploits physical topology information to perform P2P distributed data reconciliation, a major function for collaborative applications. Our solution (P2P-Reconciler-TA) relies on dynamically selecting nodes to execute specific steps of the algorithm, while carefully placing relevant data. We show that P2P-Reconciler-TA introduces a gain of 50% compared to P2P-Reconciler and still scales up.

1 Introduction

Collaborative applications are getting common as a result of rapid progress in distributed technologies (grid, P2P, and mobile computing). There are currently many projects aimed at constructing these applications on top of P2P networks because of their properties: decentralization, self-organization, scalability and fault-tolerance. As an example of such applications, consider a community of scientists working on the same project while geographically dispersed. Scientists collaborate by sharing and processing data without relying on a central server.

A P2P network is an overlay network built over a physical network. Each node is logically connected to a set of nodes, referred to as its neighbors. Normally, the neighborhood of a node is set without much knowledge of the underlying topology, causing a mismatch between the P2P overlay and the physical network. Thus, communications between nodes incur high latencies and overload the network. Distributed P2P algorithms, used by collaborative applications and built on top of P2P networks, introduce large data transfers and frequent interactions between the nodes involved. The performance of such algorithms may degrade drastically because of two orthogonal problems: inefficient overlay and nodes selection without taking into account topology information.

Approaches have been focusing on the P2P overlay network. The goal is to construct an overlay that reflects the underlying topology. Some proposals such

as [2] group nodes into clusters based on network distance or IP addresses. Others [3, 7–9] try to improve nodes neighborhood in terms of proximity. Regarding the selection issue, studies such as [7] are typically limited to finding one nearby node wrt. an origin node when searching for data replicated at multiple nodes. In this paper, we propose a solution at the application level (over a P2P network). We focus on P2P semantic reconciliation, used within optimistic replication.

Optimistic replication is largely used as a solution to provide data availability for dynamic collaborative applications. It allows the asynchronous updating of replicas such that applications can progress even though some nodes are disconnected or have failed. This enables asynchronous collaboration among users. However, concurrent updates may cause replica divergences and conflicts, which should be reconciled. The P2P-Reconciler approach [4] performs distributed semantic reconciliation over a P2P network structured as a distributed hash table (DHT) [6, 8]. Thus, the mismatch between the P2P overlay and the physical network may introduce poor performance.

In this paper, we propose P2P-Reconciler-TA, an approach used to improve P2P-Reconciler response times. P2P-Reconciler-TA dynamically takes into account the physical network topology combined with the DHT properties when executing reconciliation. Our approach is designed for distributed reconciliation. However the metrics, cost functions and the general approach can be useful in different contexts.

The main contributions of this paper are: (1) metrics and cost functions that rely on characteristics of both P2P and underlying networks; (2) a distributed algorithm for dynamically selecting the best nodes to execute specific reconciliation steps while considering dynamic data placement; and (3) experimental results that show that P2P-Reconciler-TA yields excellent scalability, with very good performance and limited overhead.

The rest of this paper is organized as follows. Section 2 discusses the most relevant related work. Section 3 provides preliminaries that constitute the basis of our work. Section 4 presents P2P-Reconciler-TA, detailing our metrics, cost functions and topology-aware approach. Section 5 gives a performance evaluation based on our implementation. Section 6 concludes.

2 Related Work

Many efforts have been made to exploit topological information in order to improve the performance of P2P environments. In [3], the authors describe a measurement-based technique to dynamically connect physically close nodes and disconnect physically distant nodes. A design improvement of the P2P overlay CAN [6] aims at constructing it in a way congruent to the underlying topology. Pastry [8] and Tapestry [9] are both P2P overlay routing infrastructures that take into account network locality to establish nodes neighborhoods. In comparison with these works, we focus mainly on the algorithms, not on the overlay structure. Given a set of nodes, we exploit topological information to select the "best" nodes to participate in the different steps of an algorithm, in a

way that achieves optimal performance. As such, those approaches can be used to complement our work and improve our results.

Location-aware clustering is proposed in [2] where physically close nodes are grouped into clusters. However, the control of the topology relies on a centralized server, which introduces high overheads due to the dynamic changes which are frequent in P2P environments. In contrast, our solution is based on network information gathered and refreshed in a scalable and inexpensive manner.

3 Preliminaries

In this section, we introduce some preliminaries that constitute the basis of our work. We briefly describe the P2P-Reconciler and CAN necessary to understand our topology-aware approach.

3.1 P2P-Reconciler

P2P-Reconciler [4] performs distributed semantic reconciliation over a P2P network structured as a distributed hash table (DHT) [6, 8]. A DHT provides a hash table abstraction over multiple computer nodes. Data placement in the DHT is determined by a hash function which maps data identifiers into nodes.

P2P-Reconciler takes advantage of the action-constraint framework of IceCube [1] to perform semantic reconciliation. According to this framework, the application semantics can be described by means of constraints between actions. A *constraint* is an application invariant, e.g. *a parcel* constraint establishes the "all-or-nothing" semantics, i.e. either all *parcel*'s actions execute successfully in any order, or none does. For instance, consider a user that improves the content of a shared document by producing two *related* actions a_1 and a_2 (e.g. a_1 changes a document paragraph and a_2 changes the corresponding translation); in order to assure the "all-or-nothing" semantics, the application should create a parcel constraint between a_1 and a_2. These actions can conflict with other actions. Therefore, the aim of reconciliation is to take a set of actions with the associated constraints and produce a schedule, i.e. a list of ordered actions that do not violate constraints.

With P2P-Reconciler, reconciliation is executed in 6 distributed steps in order to maximize parallel processing. Each step is performed simultaneously and independently by a subset of nodes referred to as *reconciler nodes*. Data produced or consumed during reconciliation are held by different *reconciliation objects*. A reconciliation object is stored in one particular node called *provider node*, based on its object identifier. We restrict the reconciliation work to a subset of nodes (the reconciler nodes) in order to maximize performance.

P2P-Reconciler's steps proceed as follows. First, nodes execute local actions to update replicated data while respecting user-defined constraints. Then, these actions and constraints are inserted into the appropriate reconciliation objects. When the reconciliation is launched, P2P-Reconciler selects the best reconcilers according to communication costs. Once reconcilers are chosen, they retrieve

actions and constraints from their corresponding provider nodes and produce a global schedule by resolving conflicting updates based on the application semantics. This schedule is locally executed at every node; thereby assuring eventual consistency (i.e. all replicas eventually achieve the same final state when users stop submitting updates).

3.2 CAN

Basic CAN [6] is a virtual Cartesian coordinate space to store and retrieve data as *(key, value)* pairs. At any point in time, the entire coordinate space is dynamically partitioned among all nodes in the system, so that each node owns a distinct zone that represents a segment of the entire space. To store (or retrieve) a pair (k_1, v_1), key k_1 is deterministically mapped onto a point P in the coordinate space using a uniform hash function. Then (k_1, v_1) is stored at the node that owns the zone to which P belongs. Intuitively, routing in CAN works by following the straight line path through the Cartesian space from source to destination coordinates.

Optimized CAN aims at constructing its logical space in a way that reflects the topology of the underlying network. It assumes the existence of well-known landmarks spread across the network. A node measures its round-trip time to the set of landmarks and orders them by increasing latency (i.e. network distance). The coordinate space is divided into bins such that each possible landmarks ordering is represented by a bin. Physically close nodes are likely to have the same ordering and hence will belong to the same bin.

4 P2P-Reconciler-TA

P2P-Reconciler-TA is a distributed protocol for reconciling conflicting updates in *topology-aware* P2P networks. A P2P network is classified as topology-aware if its topology is established by taking into account the physical distance among nodes (e.g. in terms of latency times). P2P-Reconciler-TA aims at exploiting the physical proximity of nodes to improve the reconciliation performance. Briefly, P2P-Reconciler-TA works as follows. Based on the network topology, it selects the best provider and reconciler nodes. These nodes then reconcile conflicting updates and produce a schedule, which is an ordered list of non-conflicting updates. In this work, we focus on node allocation by proposing a dynamic distributed algorithm for efficiently selecting provider and reconciler nodes. We first introduce some definitions. Then, we present the allocation algorithm in details.

4.1 Metrics

P2P-Reconciler-TA uses the following reconciliation objects: action log (L_R), action summary (AS), clusters set (CS), and schedule (S). The action log contains update actions to be reconciled; the action summary holds constraints among actions; the clusters set stores clusters of conflicting actions; and the schedule

holds an ordered list of actions that do not violate constraints. For availability reasons, we produce k replicas of each reconciliation object and store these replicas into different providers. We note these terms as follows:

- **RO**: set of reconciliation objects $\{LR, AS, CS, S\}$
- **ro**: a reconciliation object belonging to RO (e.g. CS, LR, etc.)
- $\mathbf{ro_i}$: the replica i of the reconciliation object ro (e.g. CS_1 is the replica 1 of CS), where $1 \leq i \leq k$; the coordinates (x_i, y_i) are associated with ro_i and determines the ro_i placement over the CAN coordinate space; ro_i is stored at the provider node p_{roi} whose zone includes (x_i, y_i)
- $\mathbf{P_{ro}}$: set of k providers p_{roi} that store replicas of the reconciliation object ro
- $\mathbf{best}(\mathbf{P_{ro}})$: the most efficient provider node holding a replica of ro

We apply various criteria to select the best provider nodes. One of these criteria establishes that a provider node should not be isolated in the network, i.e. it should be close to a certain number of neighbors that can become reconcilers, and therefore are called *potential reconcilers*. The physical proximity in terms of latency is not enough; a potential reconciler should also be able to access provider's data at an acceptable cost. Thus, such a potential reconciler is considered a *good neighbor* of the associated provider node. We now present metrics and terms applied in provider node selection:

- **accessCost(n, p)**: cost for a node n accessing data stored at the provider node p in terms of latency and transfer times. The transfer time relies on the message size, which is usually variable. For simplicity, we consider a message of fixed size (e.g. 4 Kb). Equation (1) shows that the $accessCost(n, p)$ is computed as the latency between n and p (noted $latency(n, p)$) plus the time to transfer the message msg from p to n (noted $tc(p, n, msg)$

$$accessCost(n, p) = latency(n, p) + tc(p, n, msg) \tag{1}$$

- **maxAccessCost**: maximal acceptable cost for any node accessing data stored in provider nodes; if $accessCost(n, p) > maxAccessCost$, n is considered far away from p, and thus is not a good neighbor of p
- **potRec(p)**: number of potential reconcilers that are good neighbors of p
- **minPotRec**: minimal number of potential reconcilers required around a provider node p in order to accept p as a candidate provider; if $potRec(p) < minPotRec$, p is considered isolated in the network
- **candidate provider**: any provider node p with $potRec(p) \geq minPotRec$ is considered a candidate in the provider selection
- **cost provider**: node that stores costs used in node selection.
- **QoN(p)**: quality of network around the provider node p. It is defined as the average access cost associated with good neighbors of p, and it is computed by (2). In this equation, n_i represents a good neighbor of p

$$QoN(p) = \frac{1}{potRec(p)} \sum_{i=1}^{potRec(p)} accessCost(n_i, p) \tag{2}$$

Another criterion for selecting a provider node is its proximity of other providers. During a reconciliation step, a reconciler node often needs to access various reconciliation objects. By approximating provider nodes, we reduce the associated access costs. We also need the following terms applied in reconciler selection:

- **candidate reconcilers**: set of nodes that are candidate to become reconcilers. This set includes all good neighbors of selected providers
- **step**: a reconciliation stage

4.2 Detailed Algorithm

P2P-Reconciler-TA selects provider nodes and candidate reconcilers as follows. Every provider node regularly evaluates its network quality and, according to the number of potential reconcilers around it, the provider announces or cancels its candidature to the cost provider node. The cost provider, in turn, manages candidatures by monitoring which providers have the best network quality. Whenever the best providers change, the cost provider performs a new selection and notifies its decision to provider nodes. Following this notification, provider nodes inform their good neighbors whether they are candidate reconcilers or not. With the selection of new providers, current estimated reconciliation costs are discarded and new estimations are produced by the new candidate reconcilers. Thus, selected provider nodes and candidate reconcilers are dynamically changing according to the evolution of the network topology. We now detail each step of node allocation.

Computing Provider Node's QoN. A provider node computes its network quality by using equation (2) and the input data supplied by its good neighbors. Good neighbors introduce themselves to the provider nodes as follows. Consider that node n has just joined the network. For each reconciliation object $ro \in RO$, n looks for the closest node that can provide ro, noted p_{ro}, and if $accessCost(n, p_ro)$ is acceptable, n introduces itself to p_{ro} as a good neighbor by informing $accessCost(n, p_{ro})$. Node n finds the closest p_{ro} as follows. First, n uses k hash functions to obtain the k coordinates (x_i, y_i) corresponding to each replica ro_i. Then, n computes the Cartesian distance between n's coordinates and each (x_i, y_i). Finally, the closest p_{ro} is the one whose zone includes the closest (x_i, y_i) coordinates. The closest p_{ro} is called the n's *reference provider* wrt. ro. Figure 1 illustrates how node n finds its reference provider wrt. the action summary reconciliation object (AS).

Provider nodes and the associated potential reconcilers cope with the dynamic behavior of the P2P network as follows. A provider node dynamically refreshes its QoN based on its good neighbors' joins, leaves, and failures. Joins and leaves are notified by the good neighbors whereas failures are detected by the provider node based on the expiration of a ttl (time-to-live) field. On the other hand, a good neighbor dynamically changes a reference provider p_{ro} whenever p_{ro} gives up the responsibility for ro. If p_{ro} disconnects or transfers ro to another provider, p_{ro} notifies these events to its good neighbors. However, if

p_{ro} fails its good neighbors detect such failure and change the corresponding reference provider.

Managing Provider Candidature. The network quality associated with a provider node dynamically changes as its potential reconcilers join, leave, or fail. Thus, a provider node often refreshes its candidature as follows. When the neighborhood situation of a provider p switches from isolated (i.e. p has a few of potential reconcilers around it) to surrounded (i.e. $potRec(p) \geq minPotRec$), p announces its candidature to the cost provider. In contrast, when p switches from surrounded to isolated, p cancels its candidature. Finally, if p's QoN varies while it remains surrounded by potential reconcilers, p updates its QoN. Figure 2 illustrates AS candidate providers for $minPotRec = 4$.

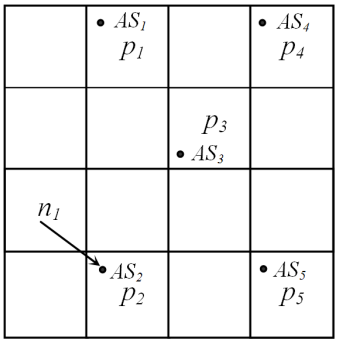

Fig. 1. Finding the AS reference provider

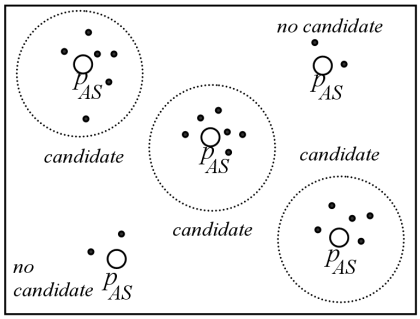

Fig. 2. Managing provider candidature

Selecting Provider Nodes. For each reconciliation object, P2P-Reconciler-TA must select the best provider node. This selection should take into account the proximity among providers since different providers are accessed in the same reconciliation step. We reduce the search space of best providers by applying the heuristic illustrated in Fig. 3. First, we select $best(P_{AS})$ and $best(P_{CS})$ (Fig. 3a). These nodes must be as close as possible from each other because AS and CS are the most accessed reconciliation objects and both are often retrieved in the same step. Next, we select $best(P_{LR})$ and $best(P_S)$ based on the pair $(best(P_{AS}), best(P_{CS}))$ previously selected (Fig. 3b); $best(P_{LR})$ must be as close as possible to $best(P_{AS})$ since a reconciler accesses both $best(P_{LR})$ and $best(P_{AS})$ in the same step whereas $best(P_S)$ must be as close as possible to $best(P_{CS})$ for the same reason. Figure 3c shows the selected providers of our illustrative scenario (i.e. p_{AS1}, p_{CS3}, p_{S1}, and p_{LR5}).

All candidate providers have at least $minPotRec$ potential reconcilers around them. However, the network quality (QoN) may vary a lot from one provider

to another. Therefore, instead of consider all candidates we begin the selection by filtering, for each reconciliation object, the k best providers in terms of QoN. Afterwards, we evaluate only the distances among these filtered candidates. For instance, in Fig. 3 only 2 candidates per reconciliation object were filtered (i.e. $\{(p_{AS1}, p_{AS3}), (p_{CS3}, p_{CS4}), (p_{S1}, p_{S2}), (p_{LR1}, p_{LR5})\}$). For selecting the pair $(best(P_{AS}), best(P_{CS}))$, the cost provider sends the set of filtered CS providers (i.e. $FCS = \{p_{CS3}, p_{CS4}\}$) to each filtered AS provider (i.e. $FAS = \{p_{AS1}, p_{AS3}\}$). Afterwards, each $p_{ASi} \in FAS$ computes the latency between p_{ASi} and each $p_{CSj} \in FCS$, noted $latency(p_{ASi}, p_{CSj})$, and returns these latencies to the cost provider in the following tuple format: $< p_{ASi}, p_{CSj}, latency(p_{ASi}, p_{CSj}) >$. The cost provider merges such tuples arranging them in ascending order of latency. Finally, the cost provider retrieves the first tuple (i.e. the one with the smallest latency) and designates the associated pair of providers (i.e. (p_{ASi}, p_{CSj})) as selected providers. The same approach is used to select $best(P_{LR})$, which should be close to $best(p_{AS})$, as well as to select $best(P_S)$, which should be close to $best(P_{CS})$.

The candidate providers filtered to participate of the provider selection vary with time. To face this dynamic behavior of candidatures, the cost provider automatically launches a new provider selection whenever the set of filtered candidates change.

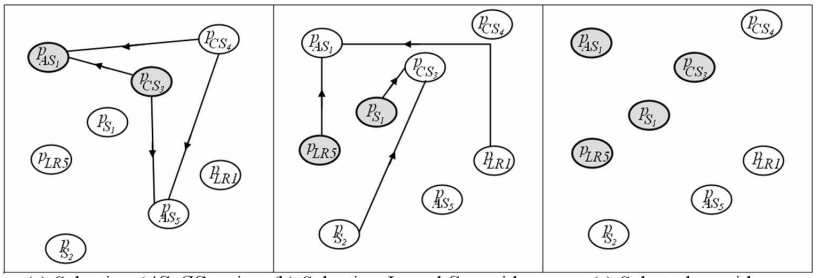

Fig. 3. Selecting provider nodes

Notifying Provider Selection Changing the selected provider leads to changes in the set of candidate reconcilers and invalidates all estimated reconciliation costs. As a result, the cost provider discards estimated costs and notifies the result of provider selection to provider nodes. The provider nodes, in turn, proceed as follows. If the provider p switches from selected to unselected, p notifies its good neighbors that from now on they are no longer candidate reconcilers. In contrast, if the provider p switches from unselected to selected, p notifies its good neighbors that from now on they are candidate reconcilers.

5 Performance Evaluation

To validate P2P-Reconciler-TA and study its performance, we implemented it on top of a simulated overlay P2P network based on topology-aware CAN. We performed many tests but, for space reasons, we present only part of the experimental results. The simulation background and our performance model are available in [5].

The main motivation for proposing P2P-Reconciler-TA is to improve reconciliation performance by taking advantage of topology-aware networks. Thus, our first experiment compares the performance of P2P-Reconciler with P2P-Reconciler-TA. Both protocols were executed in the same context (i.e. number of actions to reconcile number of connected nodes, network bandwidths and latencies, etc.). Figure 4 shows that P2P-Reconciler-TA outperforms P2P-Reconciler by a factor of 2. This is an excellent result since P2P-Reconciler is already an efficient protocol and CAN is not the most efficient topology-aware P2P network (e.g. Pastry and Tapestry are more efficient than CAN).

The second experiment aims at observing the scalability of P2P-reconciler-TA by studying the impact of the number of connected nodes on the reconciliation time (the larger the number of nodes is, the larger the average number of hops needed to lookup an identifier in the P2P network). We varied the number of connected nodes from 64 to 4000 whereas the number of reconciled actions was 106. Although most collabrative applications have limited numbers of participants, we considered large numbers of nodes to prove that P2P-reconciler-TA adapts perfectly to several other contexts. Figure 5 represents the reconciliation time with a straight line, which means an excellent scalability wrt. the number of connected nodes. This happens because provider and reconciler nodes are as close as possible independently of the network size.

Recall that for each reconciliation object, P2P-Reconciler-TA must select the best provider node. Even with a limited number of replicas, the search space is quite large as the combination of provider nodes must be taken into account. We aim at drastically reducing the search space of best providers while preserving the best alternatives in the reduced space. This allows us to efficiently select provider nodes. So, our third experiment studies the selection of provider nodes by varying the number of candidate providers per reconciliation object. The candidates are chosen according to their network quality. Figure 6 shows that P2P-Reconciler-TA achieves the best performance with small numbers of candidates (e.g. 3 or 4). This is an excellent result since the smaller the number of candidates is, the smaller the search space.

6 Conclusion

In this paper, we proposed a topology-aware approach to improve response times in P2P distributed semantic reconciliation. The P2P-Reconciler-TA algorithm dynamically takes into account the physical network topology combined with the DHT properties when executing reconciliation. We proposed metrics and

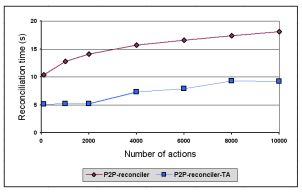

Fig. 4. Varying number of actions

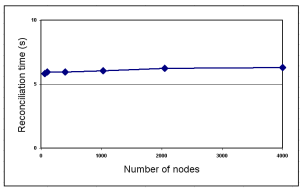

Fig. 5. Varying number of nodes

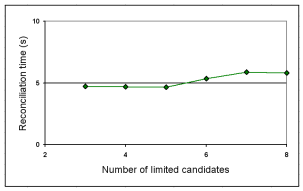

Fig. 6. Varying number of candidate providers

cost functions to be used for dynamically selecting the best nodes to execute reconciliation, while considering dynamic data placement. We validated P2P-Reconciler-TA through implementation and simulation. The experimental results show that our approach achieves a performance improvement by a factor of 2 in comparison with P2P-Reconciler. In addition, P2P-Reconciler-TA has proved to be scalable with limited overhead and thereby suitable for P2P environments. Our approach is conceived for distributed reconciliation; however our metrics, costs functions as well as our selection approach can be useful in several contexts.

References

1. Kermarrec, A-M., Rowstron, A., Shapiro, M., Druschel P.: The IceCube approach to the reconciliation of diverging replicas. Proc. of ACM PODC, 2001.
2. Krishnamurthy, B., Wang, J., Xie, Y.: Early measurements of a cluster-based architecture for P2P systems. Proc of ACM SIGCOMM, 2001.
3. Liu, Y., Xiao, L., Liu, X., Ni, L.M. Zhang, X.: Location awareness in unstructured P2P pystems. IEEE Trans. Parallel Distrib. Syst., 16(2), 2005.
4. Martins, V., Pacitti, E.: Dynamic and distributed reconciliation in P2P-DHT networks. Proc. of Euro-Par, 2006.
5. P2P-Reconciler-TA. `http://www.sciences.univ-nantes.fr/lina/gdd/members/vmartins/`
6. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. Proc. of ACM SIGCOMM, 2001.
7. Ratnasamy, S., Handley, M., Karp, R.M., Shenker, S.: Topologically-aware overlay construction and server selection. Proc. of IEEE INFOCOM, 2002
8. Rowstron, A., Druschel, P.: Pastry: scalable, distributed object location and routing for large-scale P2P systems. Proc. of IFIP/ACM Middleware, 2001
9. Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., Kubiatowicz, J.D.: Tapestry: a resilient global-scale overlay for service deployment. IEEE Journal on Selected Areas in Communications (JSAC), 22(1), 2004.