The challenges of merging two similar structured overlays: A tale of two networks*

Anwitaman Datta and Karl Aberer

anwitaman.datta@epfl.ch, karl.aberer@epfl.ch
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland

Abstract. Structured overlay networks is an important and interesting primitive that can be used by diverse peer-to-peer applications. Multiple overlays can result either because of network partitioning or (more likely) because different groups of peers build such overlays separately before coming in contact with each other and wishing to coalesce the overlays together. This paper is a first look into how multiple such overlays (all using the same protocols) can be merged - which is critical for usability and adoption of such an internet-scale distributed system. We elaborate how two networks using the same protocols can be merged, looking specifically into two different overlay design principles: (i) maintaining the ring invariant and (ii) structural replications, either of which are used in various overlay networks to guarantee functional correctness in a highly dynamic (membership changes) environment.
Particularly, we show that ring based networks can not operate until the merger operation completes. In contrast, from the perspective of individual peers in structurally replicated overlays there is no disruption of service, and they can continue to discover and access resources that they could originally do before the beginning of the merger process, even though resources from the other network become visible only gradually with the progress of the merger process.

1 Introduction

In the recent years there has been an increasing trend to use resources at the edge of the network - typically desktop computers interconnected across the internet provide services and run applications in a peer-to-peer manner, as an alternative to the traditional paradigm of using dedicated infrastructure and centralized coordination and control. Many peer-to-peer applications need some basic functionalities, particularly that of locating resources efficiently in a distributed large-scale environment in a decentralized manner. *Structured overlay*

* The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322 and was (partly) carried out in the framework of the EPFL Center for Global Computing and supported by the Swiss National Funding Agency OFES as part of the European project Evergrow No 001935.

networks have come to be recognized as a generic substrate which can facilitate resource discovery in a decentralized fashion [3, 9, 12–14, 16–18].

Even while the peer-to-peer research community prides itself to be pushing the limits of networked distributed systems, it has so far ignored a fundamental and realistic problem for structured overlays that any distributed system needs to deal with - that of making two partitions of such a system merge to become one. One can speculate several reasons for such omissions in structured overlay network research. (i) Merger of isolated overlays is trivially resolved in unstructured overlays, which is where most of the empirical information of P2P research so far has been derived from. (ii) Until recently, there has not been any real structured overlay implementations deployed and hence the problem not identified. (iii) The recent deployments and experiments have typically [1, 15] been under a controlled setting, where some central coordination like the use of a common set of bootstrap nodes has been used with the intention and sufficient coordination to construct only one overlay, making sure that independent and distinct overlays are not created. Moreover, none of these experiments with real implementations looked specifically for, or even accidentally, encounter network partitioning problems.

Apart network partitioning which can lead to the creation of two disjoint overlay networks there is a more likely scenario. It may so happen that disjoint overlay networks (using the same protocols) are formed over time by disjoint group of users. One may imagine that an overlay P2P network caters to a specific interest group from a particular geographic area who participate in an overlay network. At a later time, upon discovering a hitherto unknown group of like-mind users from a different part of the world, who use their own "private" network (using same protocols), these two groups may want to merge their networks in order to benefit from their mutual resources (like content or knowledge). In fact, such isolated overlay networks may result because of initial isolation of groups because of various reasons including geographic, social or administrative - a large company or country, which may originally restrict their users from interacting with outsiders in the overlay, and changes the policy at a later time - or purely because of partitioning of the physical infrastructure.

Structured overlays have often been touted as a generic substrate for other applications and services. Ideally, there will be one or few such universal overlays [5] which will be used by a plethora of other P2P applications. Realizing such an universal service too will need the possibility to merge originally isolated networks. Small overlays can be built independently, which may later be all merged together incrementally into a single overlay network.

One can thus imagine isolated islands of functional overlays catering to their individual participants. Someday, some member from one of these overlays may discover a member from another overlay. The natural thing to do then would be to merge the two originally isolated overlays into a single overlay network. In simple file-sharing networks, the motivation of doing so will be to make accessible content from both the networks to all the users. Similar conclusions can be drawn for various other conceivable applications of overlay networks.

In unstructured overlay networks (like Gnutella), merger of two originally isolated overlays happens trivially. Whichever peers from two originally isolated networks come in contact with each other need to establish mutual neighborhood relationship, and then onwards just need to forward/route messages to each other as they do with all other neighbors. That's all! Likewise, hierarchical (super-peer based) unstructured overlays also merge together trivially. This is because no peer has any specific responsibility and can potentially be responsible for any and all resources in the network.

Recent years have seen an increasing advocacy of structured overlays, because of the efficiency and completeness[1] their guarantees. There has been intensive study of structured overlays, looking primarily in three important aspects of such an infrastructure - (i) the topology of the routing network and the corresponding routing algorithms, (ii) resilience of such network under churn (membership dynamics) and (iii) load-balancing. These are critical issues that needed to be addressed in order to build practical structured overlay networks which can be deployed over the internet. The last five years of overlay related research concentrated on these issues. However, the issue of merger of structured overlays has so far not only not been addressed but has even hardly been recognized,[2] possibly because of the preoccupation of the P2P community with the other above mentioned infrastructure issues, which were necessary even to begin with development of actual software that could be deployed.

In this paper we take a first look at how and whether such merger can be achieved. And at what cost in terms of algorithmic complexity and development, as well as in terms of operational cost (bandwidth) and performance (interruption of service).

We do a case study for two structured overlays - ring based overlay (like Chord) and P-Grid, in order to identify the properties of an overlay network which would facilitate or hinder successful merger of distinct (but using the same protocols) overlay networks. These two networks, apart from being two of the few structured overlays which have really been implemented, benchmarked and deployed, also are representative of two very different design principles. Chord relies on the principle of maintaining what is called the ring invariant in order to guarantee functionality of the overlay network in presence of membership dynamics (churn). Maintaining the ring under churn is relatively straightforward and provides good resilience [8]. P-Grid uses prefix-based routing (PRR [13] topology). Such topology has been shown to have poorer static resilience than a ring based topology [8]. P-Grid alleviates the problem, and still avoids the use of maintaining a ring by using a different way to provide redundancy - which we call *structural replication*. This has been shown to provide very good dynamic resilience [2].

---

[1] Recall in terms of information retrieval terminology.

[2] The technical report version of the original Chord proposal makes a passing remark on the merger of isolated overlays. SkipNet [9] addresses a very special case of network partitioning which is not usable in the general case.

In a ring, there is a strict notion of ordering among all the peers, and the key-space partition these peers are responsible for. This strict ordering is exploited in defining the overlay topology and keeping it connected and to guarantee routing. In contrast, structural replication explicitly allows multiple peers to be responsible for the same key-space partition.

By ***structural replication*** we mean that (i) multiple peers can be responsible for the exactly same key space partition, i.e., these peers are mutual replicas; and (ii) each peer has multiple (functionally redundant) routing references which reduce the distance between source and destination by the same magnitude (probabilistically). CAN [14] uses similar key-space partition replication and calls it zone replication.

2 Background

In recent years the concept of structured overlays have attracted a lot of attention because of its potential to become a generic substrate for internet scale applications - as diverse as locating resources in a wide area network in a decentralized manner, address independent and robust and flexible (group) communication - e.g., application layer multicast and internet indirection infrastructure and content distribution network to name a few.

Structured overlay networks comprise of the three following principal ingredients:

(i) Partitioning of the key-space (say an interval or circle representing the real number between the range $[0, 1]$) among peers, so that each peer is responsible for a specific key space partition. By being responsible, we mean that a peer responsible for a particular key-space partition should have all the resources which are mapped into keys which are in the respective key-space partition.

(ii) A graph embedding/topology among these partitions (or peers) which ensures full connectivity of the partitions, desirably even under churn (peer membership dynamics), so that any partition can be reached from any partition to any other - reliably and preferably, efficiently.

(iii) A routing algorithm which enables the traversal of messages (query forwarding), in order to complete specific search requests (for keys).

Various applications can use transparently the (dynamic) binding between peers and their corresponding key-space partitions as provided by the overlay for resource discovery and communication purposes in a wide area network.

A structured overlay network thus needs to meet two goals to be functionally correct:

(i) *Correctness of routing*: Starting from any peer, it should be possible to reach the correct peer(s) which are responsible for a specific resource (key).

(ii) *Correctness and completeness of keys-to-peers binding*: Any and all peers responsible for a particular key-space partition should have all the corresponding keys (and none other).

Correctness of routing is achieved by maintaining the peers' routing tables correctly and using a proper routing algorithm. Correctness and completeness

of binding is achieved by moving the corresponding keys (content) as and when the partition a particular peer is responsible for changes, and synchronizing the content among replica peers.

Most structured overlays use the ring topology, or a hybrid one [16], relying on a strongly connected ring for functional correctness of routing in the overlay, while the rest of the connections among peers provide optimization, i.e., efficiency. Replication is done in immediate neighbors on the ring and hence is deterministic as long as the ring is maintained correctly. This ensures the correctness (and completeness) of keys-to-peers binding. Hence, in our case study, one candidate we consider is the Chord [17] overlay - which not only pioneered the ring topology in the context of structured overlays, but also is one of the most extensively studied and developed system.

Both because of the well developed algorithms of Chord to maintain the ring topology, as well as the relative ease in doing so and better static resilience than other topologies [8] of the ring topology, it is predominantly used in other overlays.

We'll show that when it comes to merger of two networks, the reliance on the ring for functional correctness of the overlay is in fact a liability - hard to achieve - very slow and costly in terms of communication complexity and in terms of the required coordination. Merger of two ring based networks disrupt completely the operations of the overlay, and hence the functioning of other applications and services using it.

The other overlay we consider is the P-Grid [3] network, which apart using prefix based routing [13] uses an (almost) unique feature - *structural replication* - in order to provide resilience against churn [2], instead of relying on the ring invariant used by most other overlays. With the use of structural replication, the correctness of routing is never violated even under network mergers. However completeness of keys-to-peers binding is harder to achieve. Thus merger of structurally replicated overlays is graceful because the applications running on top of the original overlays will always have access to all the resources they had access to in their isolated overlays, but discovery (and hence, access) to resources from the originally other overlay will be possible only when the peers (replicas) for the corresponding key-space partition(s) have synchronized.

We'll like to emphasize that this case study is not a quantitative evaluation to come out with a final verdict on any specific overlay - each of Chord and P-Grid networks have many nice, sometimes complimentary features as well as shortcomings - making each better suitable for different application requirements. The essential goal here is to explore the design space to better identify the features of overlay networks that can either facilitate or hinder merger of overlays - and hence get a better insight for (re-)designing such systems.

3 Network merger case-study: Chord

3.1 Chord (Recapitulation)

Chord uses SHA-1 based consistent hashing to generate m-bit identifier for each peer p, which is mapped onto a circular identifier space (key-space). Irrespective of how the peers' identifiers and keys are generated in a ring based topology, what is essential is that the peer identifiers are distinct. Similarly, unique keys are generated corresponding to each resource. Each *key* on the key-space is mapped to the peer with the least identifier greater or equal to *key*, and this peer is called *key*'s successor. Thus to say, this peer is responsible for the corresponding resource.

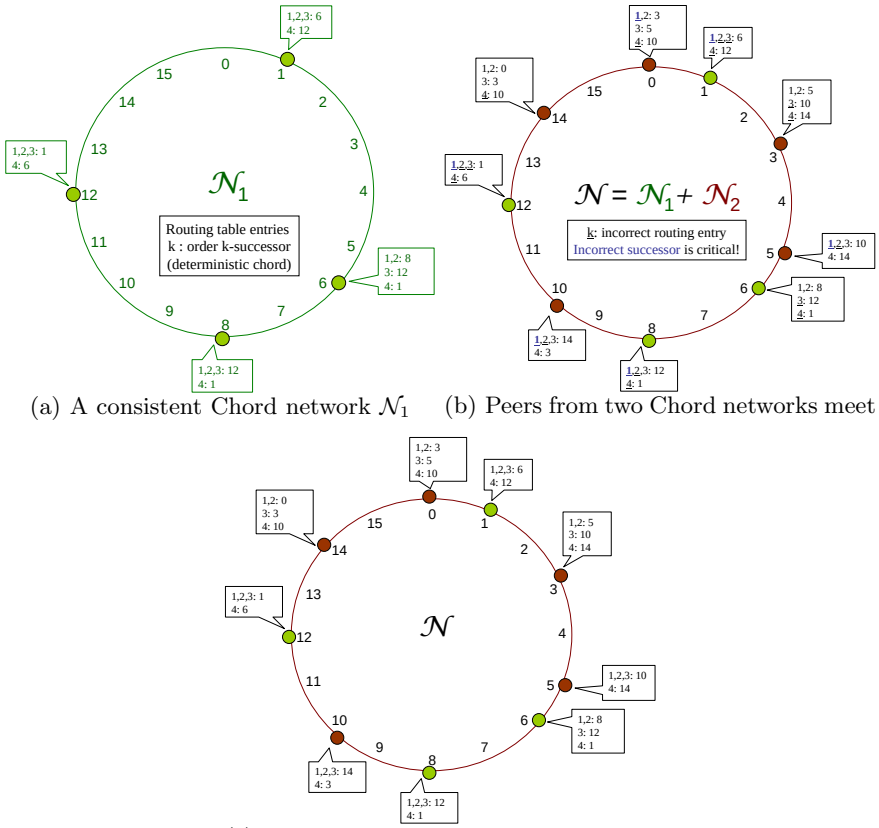
What is relevant for our study is how keys from the key-space are associated with some peer(s) and how the peers are interconnected (in a ring) and communicate among themselves.

Definition 1 *A ring network is (1) weakly stable if, for all nodes p, we have predecessor(successor(p)) = p; (2) strongly stable if, in addition, there exists no peer s on the identifier space where $p < s < q$ where successor(p) = q; and (3) loopy if it is weakly but not strongly stable.*

Condition (2) that there exists no peer s on the identifier space where $p < s < q$ if p and q know each other as mutual successor and predecessor determines the correctness of the ring structure. Figure 1(a) shows one such consistent ring structure (peer's position in the ring and its routing table). The order-1 successor known also just as "successor" of each peer is the peer closest (clock-wise) on the key-space.

If at any time such a s joins the system, the successor and predecessor information needs to be corrected at each of p, q and s. Maintaining the ring is basically to maintain the correctness of successors for all peers - this in turn provides the functional correctness of the overlay - i.e., successor peer for any identifier *key* can be reached from any other peer in the system (by traversing the ring). For redundancy, f_s consecutive successors of each peer is typically maintained, so that the ring invariant is violated only when any f_s consecutive peers in the identifier space all depart the system before a ring maintenance mechanism - Chord's self-stabilization algorithm - can amend for the changes.

In addition to the successor/predecessor information, each peer maintains routing information to some other distant peersin order to reduce the communication cost and latency. It is the way these long ranges are chosen which differ in many ring topology networks. It has no critical impact on the functional correctness of the overlay. The original Chord proposal advocated the deterministic use of the successor of the identifier $(p + 2^{k-1})$ modulo 2^m as an order-k successor of peer p or a finger table entry.Many other variants for choosing the long range links exist - e.g., randomized choice from the interval $[p + 2^{k-1}, p + 2^k)$ or exploiting small-world topology [11, 4] to name a few.



(a) A consistent Chord network $\mathcal{N}_1$ (b) Peers from two Chord networks meet

(c) Ideal Chord network comprising peers from both networks

Fig. 1. When (peers from) two ring-based overlays meet

Ring self-stabilization highlights The ring invariant is typically violated when new peers join the network, or existing ones leave it. If such events occur simultaneously at disjoint parts of the ring, the ring invariant can easily be reestablished using local interactions among the affected peers. Note that these events do not lead to a loopy state of the network.

Apart looking into the simple violations of the ring invariant which are relatively easily solved, the original Chord proposal (technical report version) also provides mechanisms to arrive at a strongly stable network starting from a loopy network (whichsoever reason such a loopy state is reached). We summarize the results of stabilizing a loopy network here.

Any connected ring network with N peers becomes strongly stable within $O(N^2)$ rounds of strong stabilization if no new membership changes occur in the system. Starting from an arbitrary connected state with successor lists of length

$O(logN)$ if the failures rate is such that at most $N/2$ nodes fail in $\Omega(logN)$ steps then, whp, in $O(N^3)$ rounds, the network is strongly stable.

3.2 Merger of two ring based networks

Consider two Chord networks $\mathcal{N}_1$ and $\mathcal{N}_2$ with N_1 and N_2 peers respectively (e.g., shown superimposed in Figure 1(b)).

When peers from different overlays meet: When peers from the two different overlays meet each other (by whatsoever reason - accidentally or deliberately), in a decentralized setting there is no way for them to ascertain that they belong to two completely different systems. This is so because overlay construction always relies on such peer meetings to start with. As a consequence, if the peer pair that meets have identifiers such that they would replace their respective successor and predecessor, then they will indeed do that.For our example from Figure 1(b) lets say peer 1 from $\mathcal{N}_1$ meets peer 0 from $\mathcal{N}_2$. Then peer 1 will treat 0 as its new predecessor, and 0 will treat 1 as its new successor, instead of 12 and 3 respectively. However, if they only change their local information, then the ring network will no more be strongly stable (may in-fact not even be weakly stable). In-fact such a reconfiguration will need and lead to a cascading effect, so that all members of both the original network try to discover the appropriate immediate neighbors (successor/predecessor) - requiring coordination among all the peers.

Estimation of the probability that a peer's predecessor changes: From the perspective of any peer in $\mathcal{N}_1$, the successor will change, if at least 1 out of the N_2 peers have identifier within the next $1/N_1$ stretch of the key-space (for which its present successor is responsible, on an average). Any particular peer from $\mathcal{N}_2$ has an identifier for this stretch with probability $1/N_1$. The number of peers falling in this stretch is thus distributed as $Binomial(N_2, 1/N_1)$, which approaches to a Poisson distribution with expectation N_2/N_1 as $N_2 \to \infty$. Hence, a peer from $\mathcal{N}_1$ will have its successor unchanged with probability $e^{-\lambda_1}$ where $\lambda_1 = N_2/N_1$. Thus each of the N_1 peers will have their successor node changed with a probability $1 - e^{-\lambda_1}$ i.i.d. Peers in $\mathcal{N}_2$ will be affected similarly with a parameter $\lambda_2 = N_1/N_2$ (symmetry).

Estimate of the number of peer pairs which will have their immediate neighbors (either successor and/or predecessor) changed: The number of peers whose successor will change in $\mathcal{N}_i$ is then distributed binomially $Binomial(N_i, 1 - e^{-\lambda_i})$ for i =1,2. Hence the expected number of nodes which will need to correct their successor nodes (and predecessor nodes) is $N_1(1 - e^{-\lambda_1}) + N_2(1 - e^{-\lambda_2})$.

The basic idea of how the ring can be reestablished is that when two peers from different networks meet so that they replace each other's successor and predecessor (immediate neighbor), then this information needs to be communicated to the original immediate neighbors, and the process continues.

There are several combinations of how the neighborhoods of the peers are affected after their interaction, each of which needs to be accounted for the actual network merger algorithm. Moreover, different combinations of faults (single or

multiple peers crashing or leaving) can happen during the ring merger, and these too need to be dealt with. The specifics of such algorithms, and evaluation of the actual ring network merger algorithm is currently underway.

We'd like to admit at this juncture that without proper and exhaustive evaluation of the exact algorithms for merging two ring networks, it is difficult to see whether a strongly stable ring can be directly achieved, or whether a sequence of faults during the merger of two rings can even lead to a loopy network, which would then require even more effort to converge to a strongly stable state using Chord's already existing self-stabilizing mechanisms.

Thus the back of the envelope analysis above just provides the expected lower-bound of the ring reestablishment process in terms of correction of successor/predecessors. The latency of such a process started because of two peers from the two networks will be $O(N_1 + N_2)$ even in there is no membership changes during the whole merger process - this is the time required to percolate the information that the ring neighborhood has changed and to discover the correct neighbor when peers from both the original networks are considered together.

Ring loses bearing during the merge process Above we provided a sketch of how to only reestablish the ring topology - which only guarantees the functional correctness of the routing process - i.e., the query will be routed to the peer which is supposed to be responsible for the key-space to which the queried key belongs.

Reestablishing the ring will be necessary in order to be able to query and locate even the objects which were accessible in the original network of any individual network. Hence, such a merger operation of ring topology based overlay will typically cause a complete interruption of the overlay's functioning.[3]

Managing keys on the merged ring Establishing the ring in itself is however not sufficient in an overlay network based index. In order to really find all keys (which originally existed in at least one of the two networks) from any peer in the merged network, it will still be necessary to transfer the corresponding key/value data to the "possibly" different peer which has become responsible for the new overlay. To make things worse, in a ring based network the queries will be routed to the new peer which is responsible for a key, so that even after the reestablishment of the ring itself, some keys that could be found in the original networks may not be immediately accessible, and will need to wait until the keys are moved to the new corresponding peer.

Lets consider that before the networks started merging, network $\mathcal{N}_i$ had key set $\mathcal{D}_i$ such that $|\mathcal{D}_i| = D_i$. Furthermore, if we consider that α fraction of the keys in the two networks is exclusive, that is $|\mathcal{D}_1 \cap \mathcal{D}_2| = \alpha|\mathcal{D}_1 \cup \mathcal{D}_2|$, then on

[3] We'd like to note that such a vulnerability may expose ring topologies to a new kind of "throwing rings into the ring" distributed denial of service (DDoS) attack, though the implications of such an attack and the amount of resources an adversary will require to make such a DDoS attack needs to be studied in greater detail.

an average, if a $\mathcal{N}_i$ node's successor changes, it will be necessary to transfer on an average α fraction of the data from network $\mathcal{N}_j$'s $\frac{1}{N_1+N_2}$ stretch of the key-space. Thus, on an average, the minimum[4] required transfer of unique data from members of original networks $\mathcal{N}_j$ to $\mathcal{N}_i$ will be $D_{tx}^{j\to i} = N_1(1-e^{-\lambda_1})\alpha\frac{D_2}{N_1+N_2}$.

Apart from assigning the data corresponding to a key on the key-space to the peer which is the successor for that key, ring based topologies provide fault-tolerance by replicating the same data at f consecutive peers on the ring.[5] Given the strict choice of f as neighborhood changes, the transferred data will in-fact have to be replicated at the precise f consecutive peers of the merged network, determining the actual minimal bandwidth consumption. Similarly, some of the original f replicas will need to discard the originally replicated content.

4 Network merger case-study: P-Grid

4.1 The P-Grid routing network

P-Grid divides the key-space in mutually exclusive partitions so that the partitions may be represented as a prefix-free set $\Pi \subseteq \{0,1\}^*$. Stored data items are identified by keys in $\mathcal{K} \subseteq \{0,1\}^*$. We assume that all keys have length that is at least the maximal length of the elements in Π, i.e.,

$$\min_{k\in\mathcal{K}} |k| \geq \max_{\pi\in\Pi} |\pi| = \pi_{max}$$

Each key belongs uniquely to one partition because of the fact that the partitions are mutually exclusive, that is, different elements in Π are not in a prefix relationship, and thus define a radix-exchange trie.

$$\pi, \pi' \in \Pi \Rightarrow \pi \not\sqsubseteq \pi' \wedge \pi' \not\sqsubseteq \pi$$

where $\pi \sqsubseteq \pi'$ denotes the prefix relationship. These partitions also exhaust the key-space, so to say, the key-space is completely covered by these partitions so that each key belongs to one and only one (because of exclusivity) partition.

In P-Grid each peer $p \in P$ is associated with a leaf of the binary tree, and each leaf has at-least one peer associated to itself. Each leaf corresponds to a binary string $\pi \in \Pi$, also called the *key-space partition*. Thus each peer p is associated with a path $\pi(p)$. For search, the peer stores for each prefix $\pi(p,l)$ of $\pi(p)$ of length l a set of references $\rho(p,l)$ to peers q with property $\overline{\pi(p,l)} = \pi(q,l)$, where $\overline{\pi}$ is the binary string π with the last bit inverted. This means that at each level of the tree the peer has references to some other peers that do not pertain to the peer's subtree at that level which enables the implementation of prefix routing

[4] The actual implementation of such a data transfer will need to identify the distinct data in the two networks and transfer only the non-intersecting one, in order to achieve this minimal effort. This is an orthogonal but important practical issue that any implementation will need to look into.

[5] The parameter f is a predetermined global constant determined by the system designer.

for efficient search. The whole routing table at peer p is then represented as $\rho(p)$ Moreover, the actual instance of the P-Grid is determined by the randomized choices made at each peer for each level out of a much larger combination of choices. The cost for storing the references and the associated maintenance cost scale as they are bounded by the depth of the underlying binary tree. This also bounds the search time and communication cost. Figure 2(a) shows instances of P-Grid network (peer's path and routing table). e.g., in $\mathcal{N}_1$, peers A and F are mutual replicas and are responsible for the key-space with prefix 00. Peer A's routing table comprise of peers C and D from the partition with prefix 1 and peer B from the partition with prefix 01.

Each peer stores a set of data items $\delta(p)$. For $d \in \delta(p)$ the binary key $\kappa(d)$ is calculated using an order-preserving hash function. $\kappa(d)$ has $\pi(p)$ as prefix but it is not excluded that temporarily also other data items are stored at a peer, that is, the set $\delta(p, \pi(p))$ of data items whose key matches $\pi(p)$ can be a proper subset of $\delta(p)$. Moreover, for fault-tolerance, query load-balancing and hot-spot handling, multiple peers are associated with the same key-space partition (*structural replication*). $\Re(\kappa)$ represents the set of peers replicating the object corresponding to key κ. Peers additionally also maintain references to peers with the same path, i.e., their replicas $\Re(\pi(p))$, and use epidemic algorithms to maintain replica consistency. Routing in P-Grid is greedy.

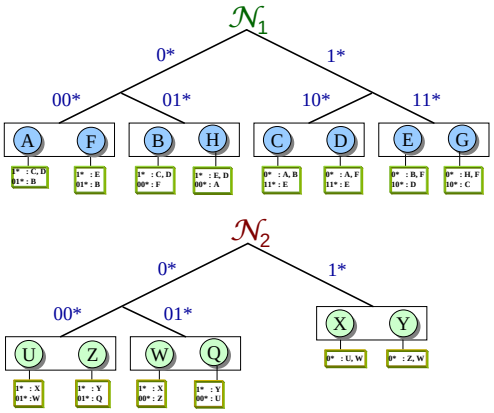
4.2 Merger of two structurally replicated networks

When peers from different overlays meet:

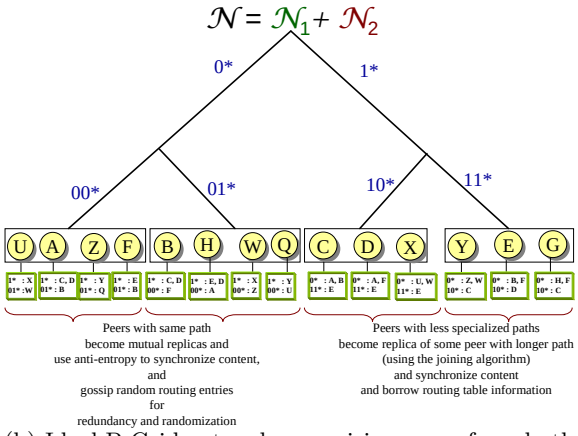
A resulting merged overlay network when peers from networks $\mathcal{N}_1$ and $\mathcal{N}_2$ meet is shown in Figure 2(b).

If peers from the two different networks meet, so that their paths are exactly the same (for example peers A and U from networks $\mathcal{N}_1$ and $\mathcal{N}_2$ respectively in Figure 2(a)), then they will execute an anti-entropy algorithm to reconcile their content and become mutual structural replicas. In fact, such an anti-entropy algorithm will have to be run among all the other structural replicas of that part of the key-space too, and eventually of the other parts as well. However, since the original members of each network still retain the original routing links, routing functionality is not affected - and whichever keys were originally accessible will continue to be accessible. So to say, peer C will always be able to access all the keys/content available at A before the merger process. The keys from the same key-space which were present in the other network would however be available only after the background replication synchronization has completed. That is to say, a resource available originally only in $\mathcal{N}_2$ at U and Z (but with the same prefix 00 as A) will be visible to C only when A has synchronized its content with any one of U or Z.

Use of structural replication has an additional downside - by not limiting the number of replicas nor having a proper structure among the replicas, it is difficult to have knowledge of the full replica subnetwork at each peer, and hence updates and replica synchronization is typically probabilistic. In contrast, once the ring is reestablished, replica positions are deterministic and hence locating replica



(a) Peers from two P-Grid networks meet



(b) Ideal P-Grid network comprising peers from both networks

Fig. 2. When (peers from) two structurally replicated overlays meet

is trivial in ring based topologies. Having discovered a replica, the anti-entropy algorithm itself (is an orthogonal issue) and hence the cost of synchronization of a pair of peers will be the same.

When two peers from $\mathcal{N}_1$ and $\mathcal{N}_2$ meet so that one's path is strictly a prefix of the other peer' path, then the peer with shorter path can execute a normal network joining algorithm [3] - extending its path to replicate either the peer it met, or a peer this peer refers it to. For example, Y may extend its path from 1 to 11. In order to do so, Y will need to synchronize its content with one of the peers which originally had the path 11, say G. Moreover Y will need to obtain routing reference to a peer responsible for the path 10 (e.g., peer C) - information it can obtain from G itself.

Since new peers join as structural replica or existing peers, no other existing peer need necessarily to update their routing table for routing functionality (unlike in a ring based topology). Thus, peer Q referring to Y for prefix 1 continues to refer to it as such, and any query 10 from Q is routed first to Y, which then forwards it to - say C. Peers may however, over time add more routing entries, for instance, Q adding a reference to D for redundancy in its routing table for the prefix 1. Such changes however is a normal process in the P-Grid network and can be carried on in the background, again without interrupting the functioning of the overlay (and in fact instead making it more resilient to faults and churn).

Consequently, neither joining peers, nor merger of two existing overlay network disrupt the available functionality of the network members.[6]

If peers with different paths meet each other, they need to do nothing, though they can refer each other to peers which are most likely to have the same path (similar to ring based topologies which can forward the peers closer to their respective key-spaces).

Managing keys in the merged network The amount of data that needs to be transferred from each system to the other is essentially the non-intersecting data. However, there is no need to transfer data from one peer to another merely because the key-space partition a peer is responsible for changes - because with structural replication, new peer joins or network mergers do not in itself automatically change the network's structure.[7]

The important thing to reemphasize is that a peer always finds the keys it could find before the merger process began, irrespective of the state of the merger process. Hence the replica synchronization can be done as a slow background process - hence the performance and network usage is also graceful - that is, merger of two overlays does not suddenly overburden the physical network's resources nor disrupt the functioning of the overlay networks. Such a graceful merger of existing networks also facilitates highly parallelized overlay construction [3] in contrast to the traditional sequential overlay construction approaches.

5 Related work

There is very little specifically looking into network partitioning issues of overlays. The only system which explicitly discusses the network partitioning issue is SkipNet [9]. SkipNet is based on ring topology and imposes a restriction on the peers' identifiers in that nodes from same administrative domain get contiguous exclusive stretch of the overlay. Thus if the domain gets partitioned, all

[6] Note that the above discussion is true only for write once and then onwards read-only data, since for read/write, it will be necessary to maintain the replicas more pro-actively.

[7] Local view of the structure however changes when a peer with shorter path meets a peer with longer path, and extends its own path according to the network construction algorithm [3], as explained above.

the nodes are contiguous on the identifier space and thus reconstructing the two rings is relatively easy. However such restrictions have serious implications on the general purposes in which the SkipNet can be used. Particularly their approach does not in any way solve the problems faced by most other overlays. Following their principles, if private overlays are first formed, either these overlays will use a minuscule portion of the key-space - which is unrealistic - particularly if its not known if and when new overlays need to be merged, otherwise, each of these private overlays will again overlap on the key-space (not be contiguous), and thus we'd again have the ring reestablishment problem as studied in this paper. Moreover, in order to find a resource, the query needs to know not only the resource name, but also the domain it can be found - which may be fine for specific applications but is very restrictive in general.

Canon [7] proposes organization of isolated overlays in a hierarchical fashion, and requires merger of the rings. However, they have not investigated the complexity of ring mergers.[8] Moreover, placement of keys in Canon is again either domain specific, so that peers from different domains may/may not find the keys. Such ambiguity (no guarantees on recall!) severely limits applications for which Canon may be used.

5.1 Network dynamics

Merger of two overlays, seen from the perspective of individual peers (which is how these decentralized systems operate) can look very similar to simple membership dynamics, churn!

However, churn is a gradual process, and the system needs to continuously perform repair operations to rectify local view of peers in order to deal with the continuous membership changes. When two isolated networks need to merge, the magnitude of the population change with respect to their original population size is very high. In fact, if we consider merger of two same-sized overlays, it'd essentially mean a vanishing half-life [10], without giving any time to the usual overlay maintenance operations to deal with the changes in the network. This also means that mechanisms other than what are employed to deal with normal churn needs to be developed. Even so, we try to reuse tools and ideas already honed in dealing with churn.

At this juncture, it is also worthwhile to point out that since merger of two networks is indistinguishable from normal churn from the perspective of individual peers, it is not obvious when to use the network merger algorithms.

5.2 Portability

Over the last few years, there has been tens (possibly even running close to a hundred) different topologies defined for structured overlays, of which some have

[8] The simulations they have seem to have merged the rings automatically. One can only speculate that most likely the new rings were formed using the global knowledge of the simulator in a manner a centralized system would. To that end, our paper exposes the complexity of achieving ring mergers.

seen sustained development and deployment - including Chord [17], Kademlia [12], Pastry/Bamboo [16, 15] and P-Grid [3]. Nonetheless, these are diverse systems using different protocols. To that end, there has been effort to identify common APIs [1, 6, 15] that can make development of applications modular and independent of the underlying overlay infrastructure.

Intercommunication of peers based on different implementations and protocols has been relatively better explored - possibly because of the proliferation of numerous overlay topology proposals as well as different implementation/protocols of theoretically equivalent networks - leading to these rather numerous tentative proposals for universal APIs.

The problem of merger of two distinct overlays (but using the same protocols) is somewhat different from how peers across different overlay networks using different protocols can communicate among themselves, or how applications can be developed and run transparently on any overlay substrate.

6 Summary and conclusion

This paper is a first step in looking at the problem of merging two separate overlay networks. Depending of the peculiarities of a specific overlay - whether it relies on the strong stability of the ring, or whether it uses structural replication, we identified the mechanisms which are necessary to execute the merger process, indicated the minimal effort it will require in order to merge two networks as well as discussed the specific challenges that need to be met for practical implementation of such an approach.

For managing data in any merged networks efficiently, it is necessary to use an efficient anti-entropy algorithm, so that ideally non-intersecting data items are identified efficiently and only those are exchanged among peers.

We also observed that the ring based networks can not function at all until the whole merge process is complete. This may have serious consequences of usability of ring based topologies, particularly if merger of two such overlay networks becomes necessary.

In contrast, use of structural replication instead of the ring as a fault-tolerance mechanism makes overlay mergers graceful. However, location of all structural replicas may be tricky, and hence replica management in such a system is relatively more complex than in ring based systems which has deterministic choice of the location and size of the replica subnetwork.

The actual algorithmic details of merger of ring based overlays is still under study and refinement, particularly looking into the potentially catastrophic combination of faults that may occur during the long latency incurred in merging two rings. Quantitative and comparative evaluation of network merger for ring based networks and structurally replicated networks, along with precise finalized algorithms to merge ring based networks thus is part of our ongoing and future work.

References

1. K. Aberer, L. O. Alima, A. Ghodsi, S. Girdzijauskas, M. Hauswirth, and S. Haridi. The essence of P2P: A reference architecture for overlay networks. In *P2P2005, The 5th IEEE International Conference on Peer-to-Peer Computing*, 2005.
2. K. Aberer, A. Datta, and M. Hauswirth. Efficient, self-contained handling of identity in peer-to-peer systems. *IEEE Transactions on Knowledge and Data Engineering*, 16(7), 2004.
3. K. Aberer, A. Datta, M. Hauswirth, and R. Schmidt. Indexing data-oriented overlay networks. *31st International Conference on Very Large Databases (VLDB)*, 2005.
4. A. Bharambe, M. Agrawal, and S. Seshan. Mercury: Supporting scalable multi-attribute range queries. In *SIGCOMM*, 2004.
5. M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. One Ring to Rule Them All: Service Discovery and Binding in Structured Peer-to-Peer Overlay Networks. In *ACM SIGOPS European Workshop*, 2002.
6. F. Dabek, B. Zhao, P. Druschel, and I. Stoica. Towards a common API for structured peer-to-peer overlays. In *IPTPS 2002*.
7. P. Ganesan, P. K. Gummadi, and H. Garcia-Molina. Canon in G Major: Designing DHTs with Hierarchical Structure. In *ICDCS*, 2004.
8. K. Gummadi, R. Gummadi, S. Ratnasamy, S. Shenker, and I. Stoica. The Impact of DHT Routing Geometry on Resilience and Proximity. In *Proceedings of the ACM SIGCOMM*, 2003.
9. N. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman. Skipnet: A scalable overlay network with practical locality properties. In *USITS 2003*, Seattle, WA, March 2003.
10. D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the Evolution of Peer-to-Peer Systems, 2002.
11. G. S. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed Hashing in a Small World. In *USITS*, 2003.
12. P. Maymounkov and D. Mazieres. Kademlia: A Peer-to-peer Information System Based on the XOR Metric. In *IPTPS*, 2002.
13. C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing Nearby Copies of Replicated Objects in a Distribute d Environment. In *SPAA*, 1997.
14. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *ACM SIGCOMM*, 2001.
15. S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. OpenDHT: A Public DHT Service and Its Uses. In *SIGCOMM*, 2005.
16. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 2001.
17. I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In *ACM SIGCOMM, (Technical report version: http://pdos.csail.mit.edu/chord/papers/)*, 2001.
18. B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-are location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, 2001.