An Extension and Cooperation Mechanism for Heterogeneous Overlay Networks

Vincenzo Ciancaglini[12⋆], Luigi Liquori[1],
Giang Ngo Hoang[23], and Petar Maksimović[124] ⋆⋆

[1] Institut National de Recherche en Informatique et Automatique, France
[2] Université de Nice Sophia Antipolis, France
[3] Hanoi University of Science and Technology, Vietnam
[4] Mathematical Institute of the Serbian Academy of Sciences and Arts, Serbia
Email: First.Last@inria.fr

Abstract. In real-world peer-to-peer applications, the scalability of data lookup is heavily affected by network artifacts. A common solution to improve scalability, robustness and security is to increase the local properties of nodes, by clustering them together. This paper presents a framework which allows for the development of distributed applications on top of interconnected overlay network. Here, message routing between overlays is accomplished by using co-located nodes, i.e. nodes belonging to more than one overlay network at the same time. These co-located nodes serve as distributed gateways, enabling the routing of requests across overlays, while keeping overlay maintenance operations local. The protocol presented here sits on top of existing overlay protocols, and gateway-node discovery can be performed either by exploiting the traffic of the overlay networks, when embedding additional data into the messages of the underlying protocols is feasible, or via active notifications independent of the underlying protocols, thus allowing an easy interaction of our protocol with already deployed overlay systems.
Keywords. Peer-to-peer, overlay networks, interoperability.

1 Introduction

Context. Overlay networks, structured and unstructured alike, have been broadly recognized as one viable solution for the implementation of various distributed applications: Distributed Hash Tables (DHTs), application-level multicast protocols, distributed object lookup services, file systems, etc. Within a real-world scenario, a fully distributed design needs to rely as much as possible on regular, desktop-class machines, without the assumption of their permanent connectivity, or even a public IP address. In contrast, one could assume that overlay networks would become even more pervasive in the future, involving smaller terminals, such as mobile devices or set-top boxes.

Despite the fact that the majority of the protocols designed during the recent years have the theoretical properties of scalability, fault tolerance and handling of dynamic topologies, studies such as [11] have shown that, under real networking conditions, overlays are still vulnerable to severe performance degradation once deployed on a larger scale. This problem has been addressed in works such as [12] for specific cases, or, for a more general case, using hierarchical overlays, in an effort to increase locality and reduce the size of individual overlays.

The second challenge with which one is faced when designing distributed applications stems from the fact that load balancing and data complexity are often inversely proportional. In the

⋆ Corresponding author.
⋆⋆ Work partially supported by the Serbian Ministry of Education and Science (projects ON 174026 and III 044006).

case of DHTs, adding semantics to overlay keys or using non-consistent hash functions (such as locality preserving hash) for the addressing implies alterations to the data partitioning scheme, resulting in an uneven distribution of data across peers. Consequently, query complexity in distributed applications would have to remain low if one were to expect low response times. Works such as [6] or [23] present the implementation of complex queries on top of structured overlays, but their approach is, again, specific to a data domain (in the above-mentioned case, semantic and "geographical" data). However, they do not address, for instance, scenarios in which an application would like to take advantage of both.

The third concern arises out of security and anonymity issues, inherently present due to the distributed nature of the system. Malicious attacks such as Sybil or Eclipse [22] should be a source of concern, when one is designing a distributed system. Furthermore, due to the peer-to-peer message routing nature of such a protocol, it becomes possible for a malicious node to infer the activity of a peer by analyzing the ongoing traffic.

Finally, there exists a more general problem regarding co-operation of various systems: at this point, it is quite difficult for different overlay networks to communicate and exchange data with each other, because of fundamental differences in their respective protocols (e.g. key- vs. keyword-based routing, recursive vs. iterative routing, exhaustive vs. non-exhaustive routing, etc.). As an example, the BitTorrent protocol [1] can use a DHT, based on the Kademlia protocol [17], to perform peer discovery: currently, there exist two different implementations of such a protocol (Mainline DHT and Azureus), and these two implementations are incompatible with one another. These incompatibilities become a real pragmatic issue in cases where one would like to merge or simply interconnect different overlay networks (or incompatible versions of the same protocol) with a huge number of peers. Often, the only possibilities are (i) an agreement between the two communities developing the overlays in question, (ii) choosing one protocol and inviting each peer to download the client, if the protocol said peer is using is different, or (iii) the blending of one overlay into the other by merging, a costly and linear operation that cannot be undone.

With this in mind, we feel that developing a strategy which involves only a simple interconnection of overlays and adheres to the "connect and leave-as-is" principle would be more economic, scalable and democratic.

Synapse. In this paper, we propose a novel framework (consisting of a related protocol, software architecture, simulator libraries and a client for peers), which we will henceforth refer to as the *Synapse Framework*, to allow for the transparent interaction between heterogeneous overlay networks. The general idea behind Synapse is the exploitation of co-located nodes, i.e. nodes connected to different overlays at the same time, as a form of *distributed gateways*, capable of inter-routing queries and messages across overlays. Nodes inside an overlay can route messages to that overlay itself, using the corresponding protocol, or to outside networks by contacting, in an unstructured way, previously discovered gateway nodes.

In [14], we have presented the first version of the protocol, where we have shown how this co-located nodes approach can bring several benefits through diversification of routing paths, the overcoming of network partitions and an increase in the locality of maintenance messages for structured overlay networks which implement key-based DHTs. Furthermore, both simulation and experimental results have shown that the routing latency of an interconnected system does not increase with regard to a single overlay with the same number of nodes. This first version, however, implemented a simple but effective opportunistic routing, i.e. a request could transit to an outside overlay only if a synapse node was routing the request in the original one. Moreover, the protocol relied on embedding additional data required for the inter-routing of the request (the non-hashed key, among others) into the overlay messages themselves, requiring a formal

change in message structure of the underlying overlay protocol. For non-collaborative scenarios, in which the overlay protocol could not have been changed due to the presence of legacy nodes, an additional "control" DHT was used to perform the inter-routing data between synapse nodes.

In the present work, we introduce several improvements over the original protocol, in order to provide a more general framework, which allows the design of routing schemes and applications based on multiple overlay networks interconnected via co-located nodes, rather than a simple routing protocol. In particular, this new system differs from and improves the one introduced in [14] in the following aspects:

- Instead of embedding sensitive inter-routing data in the overlay messages, or sharing it via a DHT, these data are now exchanged only between the requesting peer and a gateway node via a dedicated encrypted message;
- Peers can discover new gateway (co-located) nodes, thus building an unstructured overlay neighborhood on top of the structured one using various mechanisms, such as overlay message analysis, explicit notifications, or peer exchange;
- Rather than relying solely on opportunistic routing, the protocol now implements different routing strategies, using which one could send, at the same time, inter-routing requests to gateway nodes inside a peer's neighborhood and requests to structured overlays to which this peer is connected to.
- Last but not least, the framework includes an implementation of Synapse in the OverSim simulator: the latter being improved with a new feature which allows for dynamic instantiation of *multiple* and *different* overlay modules *per node*.

Because of these major improvements, what we have now is an entirely new and more general framework, capable of orchestrating multiple overlay networks, as well as transparently interconnecting existing overlays, with the possibility of designing new inter-routing schemes which may vary depending on the application or the networking conditions (e.g. better-performing request routing which can tolerate higher churn rates), up to the possibility of replicating structured routing, such as the one present in hierarchical overlay networks.

Summary. The remainder of the paper is structured as follows: in Section 2 we elaborate on the related work. In Section 3 we present the Synapse framework in detail, as well as the routing and node discovery mechanisms. Next, in Section 4 we briefly comment of some details of the implementation of Synapse in the OverSim simulator and present and discuss the results obtained therein, while in Section 5 we present and discuss the results from experiments performed on the Grid5000 platform. Next, in Section 6, we focus on several examples of applications that have already been or could be developed on top of Synapse. Finally, in Section 7 we give our conclusions and discuss further work. A web appendix containing (i) the code of the implementation of Synapse in Oversim, (ii) the modified Oversim, (iii) the code of the Synapse client, and (iv) full results of simulations and deployment can be found on the web appendix `http://www-sop.inria.fr/teams/lognet/synapse`.

2 Related work

Efficient cooperation between heterogeneous overlays has served as an inspiration to a number of research efforts, which we can classify into the following categories:

Cooperation via gateway. In [4], the authors have presented a model in which dedicated peers are used as gateways for forwarding requests from one overlay to the other. However, this model is inherently susceptible to the "single point of failure" problem: the gateways can be burdened by load and attacks. Additionally, there were no evaluations of this model offered in [4]. In [3], the

authors have proposed a protocol which makes use of co-located peers and peers whose neighbors belong to other DHTs as virtual gateways in order to facilitate cross-searching between various DHT implementations. Although the focus in [3] was placed on wireless ad-hoc networks, the authors have claimed that their protocol can be used in wired networks as well. Unfortunately, the manner in which they evaluate their protocol remains unclear.

Cooperation via super-overlay. In [20], the authors have presented SYNERGY – a super-overlay model, attempting to achieve interaction between multiple independent overlays. They have used a "global DHT" to organize the peers across different overlays into one uniform space, and the routing between different overlays is performed on one global DHT. In [13], the authors have introduced another super-overlay model for inter-overlay cooperation. In their model, certain chosen peers from more than one overlay are exported to form a SYNERGY inter-network. When the sending and the receiving peers belong to different overlays, they are exported (i.e. they join the SYNERGY network) and, in this way, they can use the SYNERGY path to route to each other.

Cooperation via hierarchy. Several works have explored the option of exploiting hierarchical structures to build overlay topologies based on the coexistence of smaller local overlay networks. CANON, which merges leaf-networks together to form one single overlay, has been introduced in [9]. In [24], the authors have presented HIERAS, an overlay network containing many other overlay networks, which are organized across different layers within this single overlay. There, each sub-overlay contains a subset of the set of all of the system peers. In [10], the authors have proposed a model which allows interconnection between various clusters of peers, by forming an overlay of super-peers chosen from these clusters.

Merging overlays. Certain works have focused on merging several overlays into one. In [4], the authors have introduced a protocol which makes it possible for one CAN-network to be completely absorbed into another. However, as mentioned before, there is a conspicuous absence of any mathematical analysis or practical evaluation in that paper. In [8], the authors provide an analysis of the problem of merging two different overlays, while in [19], the authors give an algorithm describing the merging of two ring-based overlays. Merging overlays requires modifications of the keyspace, as well as the rearrangement of keys and data, tasks which are fairly expensive in terms of time and power processing. On the other side, the merging of heterogeneous overlays still remains impossible.

Structured and unstructured hybrid overlay systems. Some works have tried to develop hybrid systems which could use the advantages of both structured and unstructured overlays. In [15], the authors have introduced a hybrid model, mixing Gnutella and DHT overlays for a file sharing application. There, the ultra peers in Gnutella form a DHT, while the search is performed using conventional flooding techniques of overlay neighbors for normal files, and through DHT queries for rare files. In [2], the authors have developed a hybrid system which uses the graph from structured overlays with data placement and data discovery strategies of unstructured overlays. This hybrid system can use either flooding or random-walk methods to locate data, and uses the structure to its advantage, ensuring that nodes are not visited more than once during one particular query. In [21], the authors have proposed a locality-aware hybrid overlay which combines the scalability and interface of a structured network with the connection flexibility of an unstructured network. There, the nodes self-organize into structured clusters based on network locality, while connections between clusters are created adaptively using random walks.

2.1 Discussion

One point in common between our contribution and the aforementioned works is the attempt to somehow increase locality in the network, by grouping peers who are determined to be "close"

according to a defined metric (usually a network distance, or a cost function) close together in the overlay as well.

Much alike the solutions which were proposed by hierarchical overlays, we are able to cluster together peers in smaller networks in order to reduce the maintenance cost and lookup time for local lookups, which can lead to faster responses in cases of local lookups. The main difference between our work and hierarchical or super-overlay systems lies in the fact that our goal is not to invent an entire new protocol, but rather to develop a complementary one, one which could interact with and make use of existing protocols in a transparent way. In this work, we present a set of APIs and routing mechanisms which could potentially interact even with overlay networks which have already been deployed and are currently operational.

We consider the co-located nodes to be an efficient mechanism to reach beyond an overlay at a reasonable cost, without bringing special-class machines as super-peers into the picture. Such nodes can be discovered and maintained by relying on the maintenance and routing messages of the overlays, without incurring a substantial additional cost.

The system decouples the routing algorithm from the rest of the system (by reducing it to a problem of choosing a gateway at each of the steps), giving way to a simple implementation of the most common unstructured routing, and the possibility of mimicking the structured mechanism elaborated on in the previous section.

By keeping the framework simple, we achieve robustness and allow for an easy implementation of self-organization strategies in order to enable and change the number of connections going from one overlay to the other.

3 The Synapse Framework

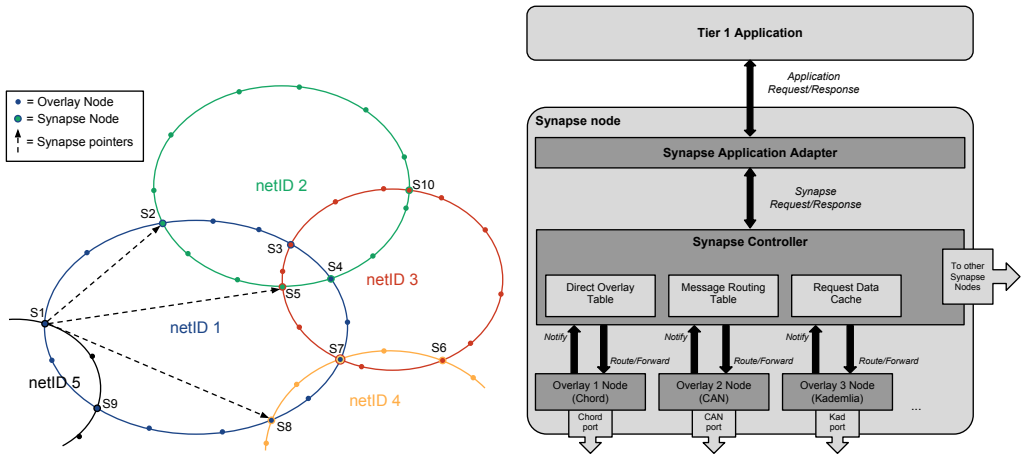
In this section, we describe the current version of the Synapse protocol, the structure of a synapse-node, and the processes of inter-overlay routing and node discovery performed in Synapse. As a reference for operations and messages, we will adopt *key-based* routing, as described in the API of [7], since there an independent and implementation-agnostic abstraction is provided, which can be used for either structured, unstructured, or hybrid overlays.

Synapse Protocol Overview. As we have mentioned earlier, the idea behind Synapse is to provide a network framework to deal with transparent interconnection and collaboration of heterogeneous overlays. Throughout the rest of the paper, we will refer to three types of nodes: *legacy-nodes*, *synapse-nodes*, and *gateway-nodes*:

- Legacy-nodes are simple instances of one single overlay protocol, they are connected to just one overlay, and are unaware of the Synapse protocol;
- Synapse-nodes are nodes which are aware of the Synapse protocol. They can be connected to one or more overlays (and, as such, they are also able to route messages in each of these overlays). Each synapse-node maintains a Direct Overlay Table (DOT) with pointers to other gateway-nodes it is aware of. The main functionalities of a synapse-node will be described later on in this section;
- Gateway-nodes are a special case of synapse-nodes, and are necessarily connected to two or more overlays.

Additionally, for each synapse-node, we will be referring to three sets of overlays related to it: *connected-overlays*, *direct-overlays*, and *indirect-overlays*:

- An overlay o is a connected-overlay of a given synapse-node s iff s is connected to o;
- An overlay o is a direct-overlay of a synapse-node s iff there exists a gateway-node s' in the DOT of s, such that o is a connected-overlay of s';



(a) Overlays with gateway-nodes (b) Structure of a synapse-node

Fig. 1: Inter-overlays topologies and synapse-nodes

– An overlay o is an indirect-overlay of a synapse-node s iff it is neither a connected-overlay of s, nor a direct-overlay of s.

Each overlay network is identified by a unique `netID`. Figure 1a shows the topology of five different overlay networks having ten synapse-nodes. Here, for instance, the connected-overlays of the synapse-node `S1` are the overlays `netID1` and `netID5`, while its direct-overlays are `netID2`, `netID3`, and `netID4` (via gateway-nodes `S2`, `S5`, and `S8`).

Hashing. Without loss of generality, we assume that all of the overlays expose key-based routing capabilities. Also, we do not require that all of the overlays use one hash-function, or are aware of all of the hashing functions. As un-hashing is not possible, routing a message outside of an overlay involves the exchange of the non-hashed key.

Synapse-node functionalities. A synapse-node must be able to:

– Process and route a message from the application layer to its connected-overlays;
– Dispatch a message directly to other synapse-nodes, to be routed in overlays others than the connected ones;
– Routing a request coming from other synapse-nodes;
– Discovering new synapse-nodes;
– Inviting new synapse-nodes to join its connected-overlays.

Apart from the data structures and messages which need to be maintained for each of the connected-overlays (finger tables, neighbors lists, routing messages, etc.), a synapse-node must handle new data structures so as to deal with inter-overlay routing:

– a *Network Identifier (netID)* per each overlay, to identify it unequivocally;
– a *Direct Overlay Table (DOT)*, which is a table in which pointers to gateway-nodes are stored, arranged per netID of the overlays they are connected to;
– a *Message Routing Table (MRT)*, responsible for storing information about ongoing messages (TTL, source nodes, RequestID, targeted overlays, etc.);

– a *Cache Table (CHT)*, used for storing values which are associated with frequently requested keys, in order to minimize routing for popular items.

Furthermore, each synapse-node should be able to interpret the following messages:

– `SYNAPSE_OFFER(netIDList)`, issued by a gateway-node in order to publish the list of overlays it is connected to;
– `SYNAPSE_REQUEST(nonHashedKey,...)`, sent by a synapse-node to a gateway-node in order to route a message outside of an overlay;
– `SYNAPSE_RESPONSE(...)`, used by a gateway-node to return response messages for a `SYNAPSE_REQUEST`;
– `SYNAPSE_INVITE(netID)`, sent to a synapse-node to "invite" it to join a specific overlay;
– `SYNAPSE_JOIN(netID)`, issued by a synapse-node wishing to join a given overlay.

Synapse routing protocol. Message routing can use three different mechanisms in synapse-nodes:

– they can route a message to any of their connected-overlays;
– they can also route a message to any of their direct-overlays by issuing a `SYNAPSE_REQUEST` message to a gateway-node in its DOT;
– they can also reach their indirect-overlays by issuing a `SYNAPSE_REQUEST` message, with the target netIDs specified, to a random set of gateway-nodes. This starts an unstructured routing mechanism through gateway-nodes until a node connected to the target overlay is found.

A `SYNAPSE_REQUEST` message carries several parameters, amongst which are the non-hashed key for the message, a RequestID (in order to identify if a message has already passed through a gateway), a TTL parameter which defines how many times should a message be routed to subsequent gateways and, if necessary, a list of target netIDs to which the message should be routed specifically. The choice of which, and how many overlays to select for message routing constitutes the routing strategy of the system.

Synapse-node routing strategies. A routing strategy consists of a set of rules which regulate the choice of overlays to which to route a message to, the choice of nodes of said overlays to route a message to, and the time instant in which a message is routed. Routing strategies strongly depend on the application implemented on top of the overlay and the network conditions. Here, we present some examples of strategies which can be implemented on top of a Synapse overlay:

– *n-Random routing:* a synapse-node picks *n* random overlays to which to route the request, out of all of its connected and direct overlay;
– *n-Flood routing:* a synapse-node picks *n* nodes per *each* direct and connected overlay. The choice of replicating a message onto the same overlay stems from the need to overcome network partitioning by routing a request through nodes placed in different locations of the addressing space;
– *n-Direct routing:* a synapse-node routes a message directly, and only to a certain overlay, by picking *n* synapse-nodes connected to said overlay. If no finger to this overlay is present, the message can be routed to random synapse-nodes by sending a `SYNAPSE_REQUEST` message with the list of target networks specified;
– *Opportunistic routing:* a synapse-node can dispatch a `SYNAPSE_REQUEST` to another synapse-node upon receipt of a `SYNAPSE_OFFER`, thus having a much higher chance of routing to an active node.

A detailed example of message routing can be found in the web appendix.

Synapse-node discovery strategies. In order to reach direct overlays, a synapse-node which joins the network needs to discover gateway-nodes connected to overlays other than his. There are several mechanisms at hand, depending on the application scenarios:

- *Message embedding (Passive Discovery):* in a collaborative scenario, in which the overlay protocol messages can support additional data, the simplest solution is to embed into the message the list of overlays the node issuing the message is connected to. In this way, each synapse-node forwarding the message in the overlay can extract this information and update its DOT, in Kademlia-like fashion;
- *Active notifications (Active Discovery):* being notified of a transiting message, a synapse-node can decide to proactively send a `SYNAPSE_OFFER` message, containing the list of its connected overlays, to the source node, in order to publish its presence. This is an effective technique in non-collaborative scenarios in which a message source is known (e.g. iterative or semi-recursive routing protocols);
- *Peer exchange:* for those scenarios in which embedding is not possible, and a message source is not known (due to a fully recursive routing algorithm), a synapse-node can still discover other synapse-nodes via an iterative peer exchange mechanism. This, however, requires an initial synapse-bootstrap-node to be contacted, in order to perform the first discovery;
- *Aggressive discovery:* apart from these strategies, which are generic and suitable for any overlay protocol, other strategies can be put in place within a collaborative scenario, to exploit specificities of a certain protocol (e.g. a source node list, leaf tables, neighbor cache, etc.).

Synapse-node structure. As shown in Figure 1b, a synapse-node consists of several components:

- The *Synapse-controller* is responsible for orchestrating multiple requests, routing messages according to the appropriate strategy, and collecting and grouping results arriving from different overlays. It also takes care of the maintenance of the synapse overlay, by performing discovery of new synapse-nodes, checking their state via ping messages, and dispatching join invitations. It maintains and relies on the DOT to store pointers to gateway-nodes and on the MRT to keep track of ongoing routings. Furthermore, the CHT can store recently retrieved values, in order to be able to serve them immediately should a new request for the same key arrive. The synapse-controller contacts its direct overlays by sending `ROUTE` messages to the overlay sub-modules. Each overlay sub-module, on the other hand, notifies the synapse-controller via a `NOTIFY` message every time an overlay message is forwarded by them or an RPC call is received, in order to check for new gateway-nodes, by examining whether or not a netID list is present in the message header, or to announce its own presence via a `SYNAPSE_OFFER` message;
- The *Synapse-application-adapter* acts as an interface to and from the application layer. It serves to decouple the applicative part from the background multi-overlay logic, by exposing an API agnostic of the underlying structure, processing complex queries, and to generate appropriate messages for the synapse-controller.

Self-organization via "social networking" primitives. In addition to Synapse messages, we propose a set of primitives which would serve to implement overlay self-organization mechanisms. By issuing a `SYNAPSE_INVITE` message, a synapse-node can propose to other synapse-nodes that they join one or more overlays, in order to, for instance, increase the overlay capacity, QoS, or external connectivity. In a similar manner, a synapse-node can offer to become a

member of an overlay, with a `SYNAPSE_JOIN` message addressed either to another synapse-node which is already a member of the target overlay, or to an authentication server.

Social-based primitives could be particularly interesting to consider in a scenario where an overlay would be able to "shrink" or "grow" around application data, such as, for instance, the social graph in online social networks. They can also be exploited to regulate connectivity of an overlay towards the rest of the system, by increasing the number of gateway-nodes to overlays in question, providing a flexible mechanism to implement QoS and failure avoidance in a system.

4 Simulation Results

In this section, we present some results obtained by running simulations within the OverSim-based Synapse simulator. In the simulations, all of the nodes were treated as synapse-nodes, and some were treated as gateway-nodes.

OverSim implementation. In order to be able to handle multiple overlay networks of different types, we have modified the OverSim simulator to support dynamic run-time instantiation of overlay modules and we have created an extension mechanism, exploiting template meta-programming of C++, to allow the existing overlay modules in OverSim to interact with the synapse-controller module, without interfering with the underlying logic.

Thanks to the common API (defined in the simulator) for communication between the application and the overlay layer, we were able to develop the synapse-controller module as a middle layer between the application and the overlay ones, receiving messages from the DHT module and sending appropriate ones to the sub-overlay modules, depending on the routing strategy, without substantial changes being made to any of the existing code.

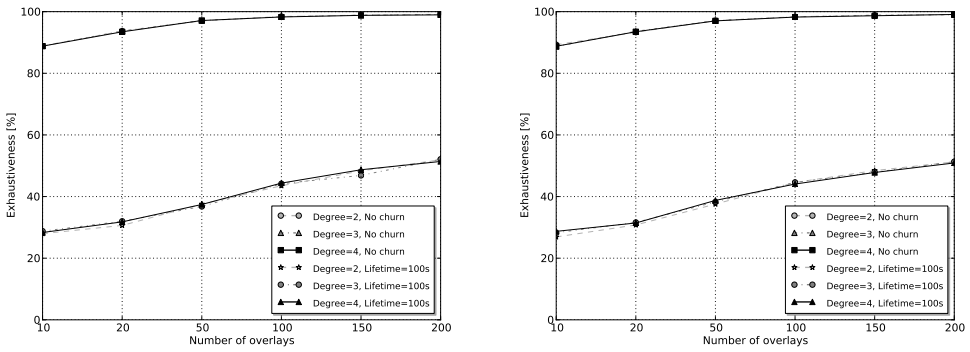
Simulation settings. Simulations were run on 2000 nodes, clustered into sub-overlays, half of which were Chord, and half of which Kademlia. All of the nodes were treated as synapse-nodes, i.e. all can perform active and passive discovery of gateway-nodes, and no legacy nodes are present.

With the system being composed of nodes clustered across many different overlays, which are connected with one another through unstructured routing, the main purpose of our simulations has been to test the reachability of each of the overlays, while varying the granularity of the network (i.e. the number of different overlays, given the same overall amount of nodes), the number of gateway-nodes present in each of the overlays and the average connection degree of gateway-nodes (i.e. the number of different overlays a gateway-node is connected to).

The tests have involved inserting random keys throughout the entire system and performing lookups for said keys, by a different node, which is not necessarily a member of the same sub-overlay in which the key is present. All replication within the sub-overlays has been disabled in order to create the most challenging conditions, and produce metrics as correlated as possible.

As the idea was to gather a lower bound of the performances of the system, we have chosen to adopt the simplest and least demanding routing strategy for synapse requests: a stateless 1-Random-Walk, meaning that nodes, at each of the routing steps, would route to every connected overlay but choose only 1 random gateway-node amongst all to reroute the request to, without considering past routing steps. Finally, the TTL has been set to 8 for all of the simulations.

We tested different scenarios, without churn, to evaluate the topology built by the node discovery process, and with high churn, i.e. with a very short node lifetime, to test it in extreme conditions, such as those of a mobile application. In all of the simulations, the connection degree was equal for all gateway-nodes. However, the percentage of gateway-nodes and their interconnection degree have been correlated to guarantee the minimum number of gateway-nodes-per-overlay to have a connected topology across all sub-overlays, without leaving any sub-overlay



(a) FIT Topology

(b) RAT Topology

Fig. 2: Effects of system granularity, with and without churn

isolated due to, possibly, a lack of gateway-nodes connected to it. The relationship between the connection degree and the number of gateway-nodes is explained in the following subsection.

Mathematical Background. Let us denote by s the number of synapse-nodes, by g the number of gateway-nodes, by d the degree of connectivity of the gateway-nodes, by o the number of overlays, and by n the overall number of nodes, calculated as $n = s + g$. Apart from these, we will require two "extended" notions: the extended number of gateway-nodes, $g_e = d \cdot g$, and the extended overall number of nodes $n_e = s + g_e = s + d \cdot s$. Using this, we can calculates:

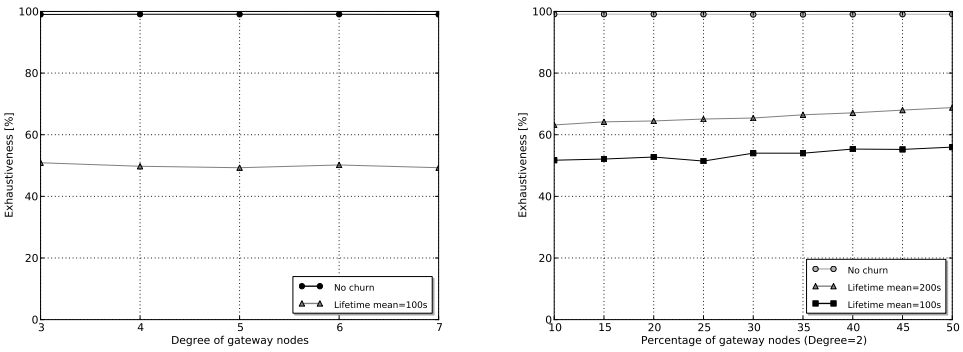
- the number of nodes-per-overlay, $n_o = \frac{n_e}{o} = \frac{l+d \cdot g}{o}$;
- the number of gateway-nodes-per-overlay, $g_o = \frac{g_e}{o} = \frac{d \cdot g}{o}$;
- the overall percentage of gateway-nodes, $s_{\%n} = \frac{g}{n}$;
- the percentage of gateway-nodes-per-overlay, $s_{\%o} = \frac{g_o}{n_o} = \frac{g_e}{n_e} = \frac{d \cdot g}{l+d \cdot g}$.

Topology construction. Topologies have been created statically, using n, o, d, and, depending on the simulation scenario, either the percentage of gateway-nodes-per-overlay, or the overall percentage of gateway-nodes in the system. Two algorithms were used to generate topologies:

1. FIT – the topology is constructed to be fully-interconnected, in the sense that from any overlay there exists a path through gateway-nodes of the system to any other overlay. This requires at least $\lceil \frac{o-1}{d-1} \rceil$ gateway nodes to be present in the system, and is accomplished using an algorithm described in the web appendix;
2. RAT – The topology is constructed with fully random assignments of overlays to gateway-nodes, using a uniform distribution over the o overlays.

4.1 Effects of system granularity

Figure 2 shows the effect that system granularity (i.e. the number of sub-overlays) has on the general system exhaustiveness. We have simulated both a churn-less environment and one with high churn, to test the topology itself, as well as its resilience to extreme conditions. Figures 2a and 2b compare a completely random topology vs. one where exhaustive connectivity has been forced. It is remarkable that the performances are substantially equivalent, suggesting that, in fact, the gateway topology can generally be built with just a partial knowledge of the system by a simple random selection of overlays. Even with 200 overlays, the routing has proved to be exhaustive, reaching every sub-overlay, and suggesting that building a clustered overlay network



(a) Few high-degree gateways

(b) Many low-degree gateways

Fig. 3: Performance comparison for different gateway topologies

is a feasible solution. The lower exhaustiveness with lower granularity is explained by the fact that, having a higher number of edges for each overlay, loops can be present, leading, with this simplest routing strategy, to requests bouncing back to the overlay they came from, an effect that can easily be avoided with a stateful routing strategy.

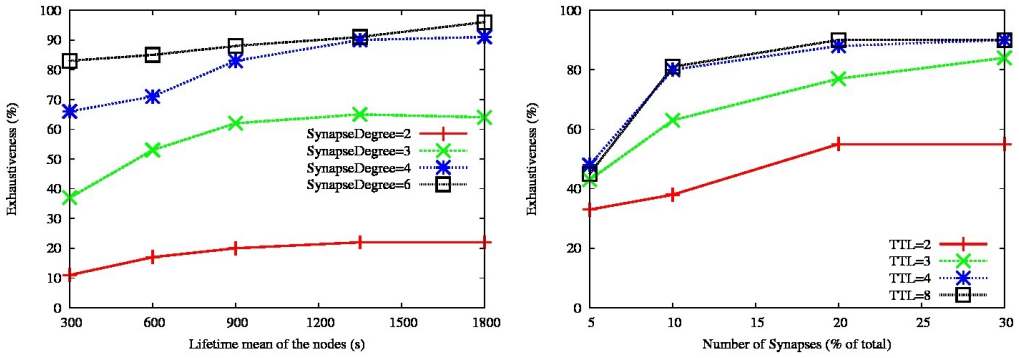
4.2 Configuration of gateway-nodes

Since maintaining a connection to multiple overlays is a costly operation, in this experiment we have tested the effectiveness of two opposite scenarios, one with very few nodes maintaining a high degree of connectivity (much like a super-peer structure), and a second one, in which an increasing number of nodes maintains a connectivity as low as possible (degree 2). It is worth noting that, despite the high connectivity degree, the gateway nodes in the first scenario were not exempted from churning.

Figure 3 shows the performances in the two scenarios. Interestingly enough, a decrease from degree 6 to degree 3 (Figure 3a) does not bring any visible decrease in performances, neither with or without churn, due partly to the simple routing strategy adopted, and it is an aspect that can be taken into account when designing a system by explicitly deploying synaps-gateways. In the second scenario (Figure 3b), on the other hand, the increase of gateway-nodes brings a slight increase in the exhaustiveness under churn, which suggests a possible strategy to handle situations of sudden churn in a system, by having most of the nodes immediately increase their connectivity degree by 1.

5 Experimental Results

In order to evaluate the behavior of Synapse within a real-world environment, we have developed a Java implementation of the Synapse protocol, which we have used to perform experiments on the national French Grid'5000 platform, that aims at providing an experimentation testbed to study large scale parallel or distributed systems which comprises thousands of interconnected computers across numerous sites in France. In all of the experiments performed, we have used 1000 nodes, distributed over 10 Chord and 10 Kademlia overlays, interconnected via the Synapse protocol.



(a) (b)

Fig. 4: Experiments on Grid5000

In the first experiment, we have investigated the exhaustiveness of the interconnected systems under different mean lifetimes of the nodes and different degrees of connectivity of synapses-nodes. We have placed an emphasis on high-churn-rate conditions (when the mean lifetime of the nodes is low), which should be observable in the near future, in overlay networks in which peers need not only be desktops and laptops, but also Internet TV and mobile devices, which are expected to join and leave the network at high frequency. In order to generate this high churn rate of nodes in the systems, we have used the Pareto distribution. The experiment was performed for mean lifetime values between 300s and 1800s, with the degree of connectivity of the synapses-nodes varying between 2 and 6, once for each of the combinations. The overall percentage of synapses-nodes was fixed to 20% of the overall number of nodes, while the TTL value was fixed to 8, in all of the cases. The results obtained from this experiment are shown in Figure 4a.

As can be seen from Figure 4a, for a fixed degree of connectivity, the Synapse protocol is fairly resilient for values of the mean lifetime above 900s, and less resilient for lower values. However, in order to achieve a sufficient level of exhaustiveness, it is necessary to increase the degree of connectivity of synapses-nodes to at least 4, for mean lifetime values above 900s, or to at least 6, for mean lifetime values below 600.

In the second experiment, we have once again investigated the exhaustiveness of the interconnected systems, this time while varying the percentage of synapses-nodes from 5% to 30%, and the TTL from 2 to 8, once for each of the combinations. The degree of connectivity of synapses-nodes has been fixed to 4, and the churn rate of the nodes to 1800s. The results obtained from this experiment are shown in Figure 4b. It can be noticed from Figure 4b that the exhaustiveness significantly increases when the TTL is increased from 2 to 4, but remains the same as the TTL is increased from 4 to 8, giving rise to the conclusion that a TTL of 4 is efficient enough when interconnected networks of this scale are concerned (20 networks, 1000 nodes overall)[5]. One other inference which can be made from Figure 4b is that having 20% of overall nodes to be synapses will result in sufficient exhaustiveness for this scale of interconnected overlays, as there is an obvious rise in exhaustiveness accompanying the increase of the number of synapses-nodes from 5% to 10% and from 10% to 20%, while no further significant rise occurs with further increase of the number of synapses from 20% to 30%.

[5] Given this result, one might question our choice of TTL in the first experiment. The reason for it being set to 8 there is the simple fact that the first experiment was performed prior to the second.

6 Application Scenarios

In this section, we will present several examples of distributed applications which could be deployed on top of the multiple interconnected overlay structure provided by the Synapse protocol, as well as some interesting aspects and possibilities which arise when considering potential applications. As expected, most of the examples take advantage of the increased level of locality offered by a federated network, be it network, geographical, social or semantic locality.

P2P Online Social Networks. In [18], the authors have described one partitioning algorithm for the Cassandra, a noSQL DHT-based database, which optimizes the key in such a manner that data which is related to users who are close in the social graph has a greater probability of being stored in the same Cassandra node. Such an algorithm could easily be adapted to a multi-overlay scenario which would be based on the Synapse protocol, enabling users to share their data using one or more DHTs, the nodes of which are links of the 1st or 2nd degree in their social graph, allowing for a scalable implementation of a real-world P2P-based social network.

Community-centric publish/subscribe. In [5], we have described a proof-of-concept of a car-sharing application based on an early design of the Synapse protocol. Such an application would present the members of a social community (e.g. school, company, etc.) with an opportunity to share service offerings (car rides, in this particular case), through the use of DHT. By using the interconnection of different DHTs, it was possible to increase the probability of matching a demand with a corresponding offer, by extending queries to other communities which are more prone to having the same travel patterns, based on geographic proximity.

Security and anonymity. Network federation can lead to the creation of trusted overlays, in which data is stored only on a set of trusted peers. Data replication across multiple overlays could lead to more efficient strategies to counter data pollution, strategies in which multiple queries for the same key could be issued across several overlays, with the idea of choosing the value which is returned by the majority of the overlays. Moreover, messages traveling within one overlay are highly unlikely to be traced back, as the request could have always come on behalf of a foreign node. The only message carrying the non-hashed key and the source node would be the `SYNAPSE_REQUEST`, which, being a point-to-point message, can be easily encrypted via a public key mechanism.

Database interoperability. In [16], we have presented another proof-of-concept, which utilizes the Synapse protocol to facilitate the interconnection of distributed digital catalogs of documents pertaining to cultural heritage. In this example, any particular document would always be under control of its owning organization, but the meta-data of the document would be shared across several DHTs, one for each of the organizations, allowing all of the users to access global data via extended queries.

7 Conclusions and Further Work

In this paper, we have presented the Synapse framework, the purpose of which is to enable the design of distributed applications based on multiple interconnected overlays, as well as to facilitate easier interconnection of already deployed overlays. The protocol has been developed in the OverSim overlay simulator, which has been modified to support multiple overlay types at run-time, and a Java client has been deployed and tested on the Grid'5000 platform. As we have just begun scratching the surface of all the possibilities offered by such an approach, our future work includes additional mathematical modeling of the protocol, an adaptation to unstructured overlays as well and further extensive testing of all the routing strategies under different system parameters, in order to be able to accurately quantify messaging overhead, resilience to churn and

data consistency. Furthermore, we will aim to define a mechanism which would guarantee only a minimum level of interconnection between different overlays, i.e. that there is a constant presence of only a minimal number of gateway-nodes within the overlays.

Acknowledgments. The authors warmly thank Erol Gelenbe for many precious discussions, and Riccardo Loti for careful reading.

References

1. Bittorrent website. http://www.bittorrent.com.
2. M. Castro, M. Costa, and A. Rowstron. Peer-to-peer overlays: Structured, unstructured, or both? Technical Report MSR-TR-2004-73, Microsoft Research, Cambridge, UK, 2004.
3. L. Cheng. Bridging distributed hash tables in wireless ad-hoc networks. In *Proc. of GLOBECOM '07*.
4. L. Cheng, R. Ocampo, K. Jean, A. Galis, C. Simon, R. Szabó, P. Kersch, and R. Giaffreda. Towards distributed hash tables (de)composition in ambient networks. In *Proc. of DSOM '06*.
5. V. Ciancaglini, L. Liquori, and L. Vanni. Carpal: Interconnecting overlay networks for a community-driven shared mobility. In *Proc. of TGC '10*.
6. P. Cudré-Mauroux, S. Agarwal, and K. Aberer. GridVine: An infrastructure for peer information management. *IEEE Internet Computing*, 11(5), 2007.
7. F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and Ion Stoica. Towards a common api for structured peer-to-peer overlays. In *Proc. of IPTPS '03*.
8. A. Datta and K. Aberer. The challenges of merging two similar structured overlays: A tale of two networks. In *Proc. of EuroNGI '06*.
9. P. Ganesan, P. Krishna Gummadi, and H. Garcia-Molina. Canon in g major: Designing dhts with hierarchical structure. In *Proc. of ICDCS '04*.
10. L. Garcés-Erice, E. W. Biersack, P. Felber, K. W. Ross, and G. Urvoy-Keller. Hierarchical peer-to-peer systems. In *Proc. of Euro-Par '03*.
11. R. Jimenez, F. Osmani, and B. Knutsson. Connectivity properties of mainline bittorrent dht nodes. In *Proc. of IEEE P2P '09*.
12. R. Jimenez, F. Osmani, and B. Knutsson. Sub-Second lookups on a Large-Scale Kademlia-Based overlay. In *Proc. of IEEE P2P '11*.
13. M. Kwon and S. Fahmy. Synergy: an overlay internetworking architecture. In *Proc. of ICCCN '05*.
14. L. Liquori, C. Tedeschi, L. Vanni, F. Bongiovanni, V. Ciancaglini, and B. Marinkovic. Synapse: A scalable protocol for interconnecting heterogeneous overlay networks. In *Proc. of Networking '10*.
15. B. Thau Loo, R. Huebsch, I. Stoica, and J. M. Hellerstein. The case for a hybrid p2p search infrastructure. In *Proc. of IPTPS '04*.
16. B. Marinković, L. Liquori, V. Ciancaglini, and Z. Ognjanović. A distributed catalog for digitized cultural heritage. In *Proc. of ICTI '10*.
17. P. Maymounkov and D. Mazières. Kademlia: A Peer-to-peer Information System Based on the XOR Metric. In *Proc. of Workshop on P2P Systems '02*.
18. J. M. Pujol, V. Erramilli, G. Siganos, X. Yang, N. Laoutaris, P. Chhabra, and P. Rodriguez. The little engine(s) that could: scaling online social networks. In *Proc. of ACM SIGCOMM'10*.
19. T. M. Shafaat, A. Ghodsi, and S. Haridi. Dealing with network partitions in structured overlay networks. *Peer-to-Peer Networking and Applications*, 2(4), 2009.
20. G. Tan and S. A. Jarvis. Inter-overlay cooperation in high-bandwidth overlay multicast. In *Proc. of ICPP '06*.
21. R. Tian, Y. Xiong, Q. Zhang, B. Li, B. Y. Zhao, and X. Li. Hybrid overlay structure based on random walks. In *Proc. of IPTPS '05*.
22. G. Urdaneta, G. Pierre, and M. Van Steen. A survey of dht security techniques. *ACM Comput. Surv.*, 43, 2011.
23. M. Varvello, C. Diot, and E. Biersack. A walkable kademlia network for virtual worlds. In *Proc. of IPTPS '09*.
24. Z. Xu, R. Min, and Y. Hu. Hieras: A dht based hierarchical p2p routing algorithm. In *Proc. of ICPP'03*.