Synapse: A Scalable Protocol for Interconnecting Heterogeneous Overlay Networks*

Francesco Bongiovanni Luigi Liquori Cédric Tedeschi
INRIA, France
surname.name@sophia.inria.fr

Bojan Marinkovic
MISANU, Serbia
bojanm@turing.mi.sanu.ac.rs

Laurent Vanni
UNSA, France
vanni@polytech.unice.fr

***Abstract*—This paper presents Synapse, a scalable protocol for information retrieval over the inter-connection of heterogeneous overlay networks. Applications on top of Synapse see those intra-overlay networks as a unique inter-overlay network.**

Scalability in Synapse is achieved via co-located nodes, i.e. nodes that are part of multiple overlay networks at the same time. Co-located nodes, playing the role of *neural synapses* and connected to several overlay networks, give a much wider search area and provide alternative routing. Built-in primitives to deal with social networking give an incentive for cooperation between nodes.

Results from simulation and experiments show that Synapse is scalable, with a communication and state cost scaling with the same complexity of the connected networks. Because of alternative routing, Synapse gives also a practical solution to network partitions.

We precisely capture the behavior of key metrics measuring Synapse and present results from simulations as well as some live run experiments of a client prototype on the Grid'5000 platform. The client implements the Synapse protocol in the precise case of the inter-connection of many Chord overlay networks.

I. Introduction

A. Context

The inter-connection of overlay networks has been recently identified as a promising model to cope with today's Internet issues such as scalability, failure recovery or routing efficiency, in particular in the context of information retrieval. Some recent researches have focused on the design of mechanisms for building bridges between heterogeneous overlay networks in the purpose of improving cooperation between networks that have different routing mechanisms, logical topologies and maintenance policies.

However, more comprehensive approaches of such inter-connections for information retrieval and both quantitative and experimental studies of its key metrics such as satisfaction rate or routing length are still missing. During the last decade, different overlay networks were specifically designed to answer well-defined needs such as content distribution through unstructured overlay networks such as Kazaa or through structured networks, mainly under the shape of Distributed Hash Tables [1]–[3], [5], publish/subscribe systems [6]–[8], networked virtual environment [9], transactional key/value store [10], or video streaming [11].

*Supported by AEOLUS FP6-IST-15964-FET Proactive: Algorithmic Principles for Building Efficient Overlay Computers and DEUKS JEP-41099 TEMPUS: Doctoral School towards European Knowledge Society.

Overlay networks' topologies span over structured, unstructured or even hybrid ones: they have completely different properties in terms of routing efficiency, exhaustivity, scalability, and reliability (refer to [12] for a survey).

B. Problematic

Many disparate overlay networks may not only simultaneously co-exist in the Internet but also compete for the same resources on shared nodes and underlying network links. One of the problems of the overlay networking area is how heterogeneous overlays networks may interact and cooperate with each others. Overlay networks are heterogeneous and basically unable to cooperate each other in an effortless way, without merging, an operation which is very costly and not suitable in many cases (security, performance, etc.).

However, in many contexts, distinct overlay networks could take advantage af cooperating for many purposes: collective performance enhancement, larger shared information, better resistance to loss of connectivity (network partitions), improved routing performance in terms of delay, throughput and packets loss, for instance by cooperative forwarding of flows. In the context of large scale information retrieval, several overlays may want to offer an aggregation of their information/data to their potential common users without losing the control of it (imagine two companies wishing to share or aggregate the information contained in their database, obviously while keeping the exclusive write to update it). Finally, in terms of fault-tolerance, cooperation can really increase the availability of the system, if one overlay becomes unavailable, the global network will only undergo partial failure as other distinct resources will be usable.

C. Contribution

The main contribution of this paper is to introduce, simulate and experiment *Synapse*, a scalable protocol for information retrieval over the inter-connection of heterogeneous overlay networks. Synapse is based on co-located nodes, also called *synapses*, serving as low-cost natural candidates for inter-overlay bridges.

In a nutshell, every message received by a co-located node can be forwarded to other overlays the node belongs to. In other words, upon receipt of a search query, in addition to its forwarding to the next hop in the current overlay (according to the routing policy of the overlay, the node can possibly

start a new search, according to some given social-based strategy, in some or all other overlay networks it belongs to. This obviously implies to provide a Time-To-Live value and detection of already processed queries to avoid infinite loop in the network, as in unstructured peer-to-peer systems. Another novelty of our design is the explicit possibility to rely on *social* primitives to build these bridges between distinct overlays. A node can *invite* another node to join its own overlay, based on some particular strategy.

Our contributions are the following:

- the introduction of *Synapse*, a generic protocol, which is suitable for inter-connecting heterogeneous overlay networks without relying on merging;
- extensive simulations in the case of the interconnection of structured overlay networks to capture the real behavior of such platforms in the context of information retrieval, identify their main advantages and drawbacks;
- the deployment of a lightweight prototype of Synapse, called `JSynapse` on the Grid'5000 platform[1] along with some real deployments showing the viability of such an approach while validating the software itself;
- finally, on the basis of the previous item, the description of a real open source prototype, called `open-Synapse`, based on the `open-Chord` implementation of Chord[2], inter-connecting an arbitrary number of Chord networks.

The final goal is to grasp the complete potential that co-located nodes offer and to deepen the study of overlay networks' interconnection using these types of nodes.

D. Outline

The remainder of the paper is organized as follows: In Section II, we summarize the mechanisms proposed in the literature related to the overlay networks' cooperation. In Section III, we introduce our Synapse protocol, by means of example and pseudocode. In Section IV, we present the results of our simulations of the Synapse protocol to capture the behavior of key metrics traditionally used to measure the efficiency of information retrieval. In Section V, we describe a deployment of a client prototype[3] over the Grid'5000 platform. Conclusions and further work conclude.

II. Related Work

In [13] and [14], authors have recently presented some architectural issues with the Internet and identified some future trends, one of which is the inter-communication of *realms*[4]. They emphasise on the inter-communication of logical networks, and discuss various ways to do so. In particular, they point out that co-existing realms should share nodes in some cases and should not in some others. For instance, networks may deliberately behave selfishly only aiming at the improvement of their application-level QoS by taking advantage of dedicated paths to other networks, i.e. using *gateways*, acting as dedicated bridges between networks. But in many cases they argue that it is more natural to take advantage of nodes that are already part of multiple overlay networks, i.e. using *co-located nodes*.

A. Cooperation through hierarchy

Pointing out the limits of a unique global structured overlay network (rigidity, maintenance cost, security, etc.), several propositions have been made over the years to build alternate topologies based on the coexistence of smaller local overlay networks. A first approach has been based on hierarchical systems [15], [16], some elected super-peers being promoted to a top-level overlay network, leading to the requirement of costly merging mechanisms to ensure a high level of exhaustiveness. In a more general view, merging several co-existing structured overlay networks has been shown to be a very costly operation [17], [18].

In the context of mobile ad hoc networks, Ariwheels [7], [19] has been designed to provide a large variety of services through a multi-layer overlay network, where super-peers, called Brokers, act as servers for a subset of peers. Ordinary peers, called Agents, submit queries to their Broker and receive results from it. Ariwheels provides an efficient mapping between physical devices in the wireless underlay network and virtual entities in the overlay network.

B. Cooperation through gateways

Authors in [20] presented two models for two overlays to be (de)composed, known as *absorption* (a sort of merging) and *gatewaying*. Their protocol enables a CAN-network to be completely absorbed into another one (in the case of the absorption), and also provide a mechanism to create bridges between DHTs (in the case of the gatewaying). They do not specifically take advantage of a simple assumption that nodes can be part of multiple overlays in the same time, thus playing the role of natural bridges.

More recently, authors in [21] propose a novel information retrieval protocol, based on gateways called *DHT-gatewaying*, which is scalable and efficient across homogeneous[5], heterogeneous[6] and assorted[7] co-existing structured overlay networks. They argue that there is not a preferred structured overlay network implementation, and that peers are members of co-existing DHTs. Their assumptions are (i) only some peers support the implementations of different DHTs and (ii) some peers are directly connected to peers that are members of other DHTs, and are called *Virtual Gateways (VG)*. When a request is sent in one overlay, and no result was found, the requester can decide to widen his search by forwarding its original search request to nodes which belong to other structured overlay networks (mapping the search to the format supported by their relative overlay). A TTL value is added

[1] http://www.grid5000.fr.

[2] http://open-chord.sourceforge.net.

[3] Freely available at http://www-sop.inria.fr/teams/lognet/synapse.

[4] A realm is a term authors use in order to identify a network instance.

[5] Same implementation and same keysize (ex. Two 160-bit Chord DHTs).

[6] Same implementation and different keysize (ex. One 160-bit Chord and one 256-bit Chord DHTs).

[7] Different implementation and/or different keysize (ex. One 160-bit Chord and one 256-CAN DHTs).

to the original search in order to avoid cycles; this value is decremented each time a request crosses a new DHT domain. Unfortunately the evaluation of their protocol lacks precious details and precision. It is unclear how they evaluate their protocol.

C. Cooperation through co-located nodes

Authors in [22] present Synergy, an overlay inter-networking architecture which improves the routing performance in terms of delay, throughput and packet loss by providing cooperative forwarding of flows between networks. Authors suggest that co-located nodes are good candidates for enabling inter-overlay routing and that they reduce traffic. Our approach can be also seen as a deeper study of their concepts.

On the way of designing inter-overlay networking based on co-located nodes, authors in [23] present algorithms which enable the symbiosis between different overlays networks with a specific application in mind: file sharing. They propose mechanisms for hybrid P2P networks cooperation and investigates the influence of system conditions such as the numbers of peers and the number of meta-information a peer has to keep. They provide interesting observations on how to join a candidate network, cooperative peers' selection, how to find other P2P networks, when to start cooperation, by taking into account the size of the network (for instance a very large network will not really benefit from a cooperation with a small network), so on and so forth. Again, a more comprehensive understanding of this approach is missing.

Authors in [24] consider multiple spaces with some degree of intersection between spaces, i.e. with co-located nodes. They focused on different potential strategies to find a path to another overlay from a given overlay, i.e. how requests can be efficiently routed from one given overlay to another one. They compared various inter-space routing policies by analyzing which trade-offs, in terms of state overhead, would give the best results, in terms of the number of messages generated and routed, the number of hops it takes to find a result and the state overhead (i.e. the number of fingers a node has to keep). They provide a comparative analytical study of the different policies. They showed that with some dynamic finger caching and with multiple gateways (in order to avoid bottlenecks and single points of failures) which are tactfully laid out, they attain pretty good performances. Their protocol focus on DHTs interconnection, we extend it to any kind of overlays.

In our previous preliminary work [25], we introduced BabelChord, a protocol for inter-connecting Chord overlay networks using co-located nodes that are part of multiple Chord "floors". These nodes connect, in an unstructured fashion, several Chord overlays together The simulations showed that we could achieve pretty high exhaustivity with a small amount of those co-located nodes. Our current paper, in turn, focuses on the co-located nodes heuristic in far more details than the aforementioned work by providing not only a more generic protocol which enables inter-overlay routing that can in principle be applied to connect arbitrary heterogeneous overlays, but also more simulations to show the behaviours of such networks as well as a real implementations and live experiments.

III. The Synapse Protocol

A. Overview

We now present our generic *meta*-protocol for information distribution and retrieval over an interconnection of heterogeneous overlay networks. Information is a set of basic `(key, value)` pairs, as commonly encountered in protocols for information retrieval.

The protocol specifies how to insert information (`PUT`), how to retrieve it through a key (`GET`), how to invite nodes in a given overlay (`INVITE`), and how to join a given overlay (`JOIN`) over a heterogeneous collection of overlay networks linked by co-located nodes. These co-located nodes represent a simple way to aggregate the resources of distinct overlays. We assume each overlay to have its own inner routing algorithm, called by the Synapse protocol to route requests inside each overlay. We assume no knowledge about the logical topology of all the involved overlay networks connected by Synapse. To ensure the usual properties of the underlying network, we assume that communication is both symmetric and transitive. Synapse simply ignores about how routing takes place inside the overlays, Synapse only offers a mechanism to route from one overlay to another in a simple, scalable and efficient way.

One important requirement of the Synapse protocol with respect to others protocols using hash functions, is that keys and nodes' addresses circulate *unhashed* from hop to hop. Recall that hash function have no inverse: once a sought key is hashed, it is impossible its initial value and this impossible to forward to another overlay having a different hash function, since hash functions can be different (in implementations and keysize) from overlay to overlay.

Obviously, both the hashed and the *clear* key data can be carried in the message or a fast hash computation can be performed at each step. Standard cryptographic protocols can be used in case of strong confidentiality requirements without affecting the scalability of the Synapse protocol itself.

More precisely, the initialization of a search query is done via the following steps: The initiator (i) logs the message in the node, (ii) sets a TTL and (iii) launches a `FIND` message in a first overlay. Upon receipt of a `FIND` message, a node checks first if the TTL is valid and second if this query was already processed on the node: in both cases, the routing aborts, in order to avoid useless message overhead.

B. Architecture and Assumptions

The inter-overlay network, induced by the Synapse protocol, can be considered as an aggregation of heterogeneous sub-overlay natworks (referred to as *intra*-overlay networks henceforth). Each intra-overlay consists of one instance of, e.g., Chord or any structured, unstructured or hybrid overlay. We recall that an overlay network for information retrieval consists of a set of nodes on which the information on some resources (collection of `(key,value)` pairs) is distributed.

```
1.01 on receipt of OPE(code,key,value) from ipsend do                    receive an operation code a key and a value from ipsend
1.02  tag = this.new_tag(ipsend);                                        create a new unique tag for this lookup
1.03  send FIND(code,ttl,mrr,tag,key,value,ipsend) to this.myip;         send a FIND message with code, ttl, mrr, tag, key, value and ipsend to itself

2.01 on receipt of FIND(code,ttl,mrr,tag,key,value,ipdest)from ipsend do   receive a FIND message with code, ttl, mrr ... ipdest from ipsend
2.02   if ttl = 0 or this.game_over?(tag)                                 the lookup is aborted because of zero ttl or because of the game over strategy
2.03   else this.push_tag(tag);                                          push the tag of the query as ''already processed''
2.04     next_mrrs = distrib_mrr(mrr,this.net_list);                     fix the associative list splitting all the mrr between all net the synapse belongs to
2.05     for all net ∈ this.net_list do                                  for all net the synapse belongs do
2.06       if this.isresponsible?(net,key)                               the current synapse is responsible for the key in the net
2.07         send FOUND(code,net,mrr,key,value) to ipdest;               send a FOUND message with code, net, mrr, key and value to ipdest
2.08       else if this.good_deal?(net,ipsend)                           the net/ipsend is a ''good'' net/synapse (left to synapse's strategy)
2.09             send FIND(code,ttl-1,next_mrr.get(net),tag,key,value,ipdest) to this.next_hop(key);   send a FIND msg with ... to ...

3.01 on receipt of FOUND(code,net,mrr,key,value) from ipsend do          receive a FOUND message with code, net, key and value from ipsend
3.02   this.good_deal_update(net,ipsend);                                update my good deal tables with net and ipsend
3.03   match code
3.04    code=GET                                                                                                  GET code
3.05     send READ_TABLE(net,key) to ipsend                             send a READ_TABLE message (omitted) through the net with key to ipsend
3.06    code=PUT                                                                                                  PUT code
3.07     if mrr < 0                                                     stop replication since no inter PUT is allowed
3.08     else send WRITE_TABLE(net,key,value) to ipsend                 send a WRITE_TABLE message (omitted) through the net with key and value to ipsend

4.01 on receipt of INVITE(net) from ipsend do                            receive an invitation to join the net from ipsend
4.02   if this.good_deal?(net,ipsend)                                    the net/ipsend is a ''good'' net/synapse (left to peer's strategy)
4.03     send JOIN(net) to ipsend;                                       send a JOIN message to reach the net to ipsend

5.01 on receipt of JOIN(net) from ipsend do                              receive a JOIN message to reach the net from ipsend
5.02   if this.good_deal?(net,ipsend)                                    the net/ipsend is a ''good'' net/synapse (left to synapse's strategy)
5.03     this.insert_net(net,ipsend);                                    the current synapse insert ipsend in the net
```

Fig. 1. The Synapse protocol

Each intra-overlay has its own key/value distribution and retrieval policy, logical topology, search complexity, routing and fault-tolerance mechanisms , so on and so forth. The Synapse protocol can be summarized by the following points:

- *Synapses:* the interconnection of intra-overlay networks is achieved by co-located nodes taking part in several of these intra-overlays, called synapses. Each peer will act according to the policy of each of its intra-overlays, but will have the extra-role of forwarding the requests to some intra-overlay it belongs to.
- *Peer's name:* every peer comes with a proper logical name in each intra-overlay; in particular, synapses have as many logical names as the number of networks they belongs to.
- *Keys mapping in peers:* each peer is responsible for a set of resources (`key,value`) it hosts. Since every intra-overlay have different policies for keys distribution, we could say that also the inter-overlay induced by Synapse inherits an homogeneous distribution among the intra- and inter-networks. As for peers, every key comes with a proper logical name peculiar to each intra-overlay.
- *Set of resources assigned to set of nodes:* all overlays protocol for information retrieval share the invariant of having a set of peers responsables of a specific set of resources. This invariant allows to route under structured, unstructured and hybrid networks: the rationale is simple: by construction, ever intra-routing is responsible for its correctness, since Synapse just care about overlay's interconnection.
- *Network independency and message translation:* intra-network protocols are different by construction: as such, when a message leaves a particular network and enters another network, the first network looses the control on the route taken inside the second one.
- *Topology, exhaustiveness, complexity and scalability:* by construction, the inter-overlay network induced by the Synapse protocol belongs to the category of unstructured overlay networks, with a routing that is not exhaustive, even if Synapse can connect only overlays that guarantee exhaustivity. The same goes for the routing complexity that can be upper-bounded only in presence of precise and strong hypotheses about the type of intra-overlay networks. The same for scalability: a Synapse inter-network is scalable if all the intra-networks are scalable.
- *Loopy routing avoidance:* to avoid lookup cycles when doing inter-routing, each peer maintains a list of already processed requests' tag in order to discard previously seen queries and a TTL value is decreased at each hop. These two features prevent the system from generating loops and useless queries, thus reducing the global number of messages in the Synapse inter-network.
- *Replications and Robustness:* to increase robustness and availability, a key can be stored on more than one peer. We introduce a Maximum-Replication-Rate (MRR) value which is decreased each time a `FIND(PUT)` message touches a synapse, thus replicating the resource in more than one intra-overlay. This action acts as a special TTL denoting how many overlays can traverse a `FIND( PUT)` message.
- *Social primitives:* each peer implements autonomously a `good_deal?` policy. This is a social-based primitive aiming at making some important choices that may strongly influence the performance and robustness of the Synapse routing. In particular, such a primitive is intended to help the choice of whether or not to join another intra-overlay,

invite or accept a peer in one of our overlay, or even create a new network from scratch. There exists no best good deal strategy: for example, if one network wants to increase connectivity with other overlays, it can suggest to all peers to invite and join all interesting/interested peers: this can be especially useful in case of high churning of the intra-network in order to increase alternative routing-path through the neighboured intra-networks.

C. Examples

We illustrate the Synapse protocol by means of two examples (From now on we denote `FIND(GET/PUT)` simply by `GET/PUT`).

1) Routing across differents intra-overlays: Figure 2 shows how a value present in one overlay can be retrieved from a `GET` launched by another overlay. Peer A in the overlay ON1 receives a `GET(key)` message: the routing goes until the synapse B, that triggers a second intra-overlay routing in ON2. The two routings proceed in parallel and in particular the routing in ON2 terminates successfully with a peer-to-peer interaction between the peer A and peer C responsible of the resource. Routing continues on ON1 until the synapse D, that triggers a third intra-overlay routing in ON3. The routing proceeds in parallel and in particular routing in ON3 terminates successfully with a second peer-to-peer interaction between A and H, while routing in ON1 proceeds to a failure on peer F via the synapse E. The synapse E launches a fourth intra-overlay routing in ON2 that proceeds to a failure on node B (game over strategy) via the synapse G. Finally, G launches a fifth intra-overlay routing on ON3, terminating with a failure on D (again game over strategy). Peers playing game over strategy are depicted as squares.

2) Dealing with network partition: Figure 3 shows how an intra-overlays can take advantage of joining each other in order to recover situations where network partitions occurs (because of partial failure of the underlay network or high peer's churn). Since network partitions affect routing performance and produces routing failures, the possibility to retrieve a value in a failed intra-overlay routing is higher thanks to the existence of alternative inter-overlay paths. More precisely the figure shows how a value stored in the peer E of the overlay ON1 can be retrieved in presence of a generic network partition by routing via ON2 and ON3 through the synapses B,C,D, and E.

D. Algorithms

Figure 1 presents the pseudo-code of the protocol using message passing paradigm.

1) The `GET` operation: The `GET` operation consists in finding the value of an object we are seeking, provided its key. A node seeking an object sends an `OPE(GET,key,_)` message to an arbitrary node it knows. On receipt (see lines 1.01-1.03), the node generates a new tag `tag` for this request that will be associated with the query all along its path. The routing is then initiated with a given TTL by sending an auxiliary `FIND` message for this request to the node itself; this message seeks

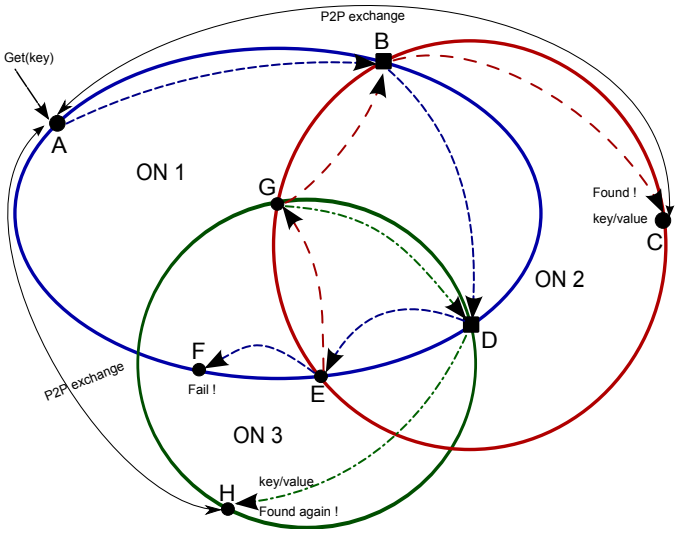

Fig. 2. Routing across differents overlays

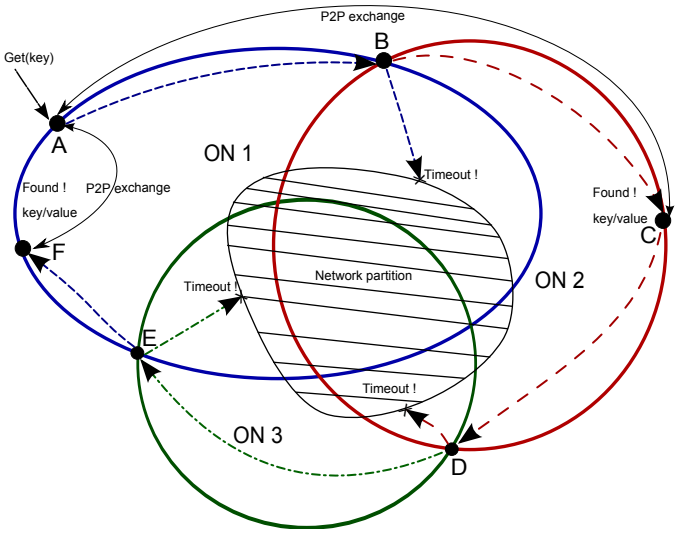

Fig. 3. Dealing with a network partition

the node(s) responsible for the key sought in order to read the value (if it exists).

On receipt of a `FIND` message (see lines 2.01-2.09), the node checks the TTL and the tag of the request before starting processing the request, and, first, recording it as *already processed* ("game over" stategy). The retrieval process starts then locally, in two steps for each intra-overlay the node belongs to: (i) it checks if, according to the particular retrieval algorithm of the intra-overlay, it is itself assigned a range of keys containing `key` (line 2.06), if this is the case, for this overlay, the retrieval process ends and a `FOUND` message is sent back to the initiator of the request informing it that the potential value sought is stored on this node (line 2.07)). (ii) if the node was not responsible for the key in this particular overlay, it forwards the request to the *next hop* inside this intra-overlay, according to the particular overlay's policy (line 2.09).

On receipt of such a FOUND message — recall that several responses can be obtained for a request — the initiator of the GET request sends a READ_TABLE message to the responsible of the key, basically to get the value of the key sought (see lines 3.04-3.05).

2) The PUT operation: The PUT operation is a declaration of a resource. Depending on the purpose of the resource aggregation, the PUT policy may change:

- If the purpose of the aggregation is to let each overlay keep the control on their information (with exclusive rights for writing and updating the information) while letting nodes from other overlay read this information, the PUT operation will be performed independently within each overlay, each node declaring their resources to their intra-overlays. In this first case, the PUT operation will not be different as in a set of intra-overlays without inter-connection, and corresponds to set the Maximum-Replication-Rate (MRR) to 0.
- If the purpose of the aggregation is to build a globally distributed information system, each node may declare its resources to a set of intra-overlays it may not belong to. In this last case, the PUT operation involves mechanisms very similar to the GET operation and the Maximum-Replication-Rate (MRR) different than zero tells how many copies we want to distribute in the inter-overlay. Line 2.04 computes via the function distrib_mrr[8] the required values of MRR for a PUT request, starting from both its current value and the number of intra-overlays the request will be forwarded to. Recall that MRR is ignored when the message is not a PUT operation. In fact, a node declaring a resource will also seek nodes in the Synapses responsible for their resources. Once such location is found (similarly than for the GET operation), the initiator of the request has just to send the value to be stored by the responsible nodes found. This is achieved by lines 3.06-3.08.

3) The JOIN and INVITE operations: The JOIN message is sent by a node entering the network, upon a reception of an INVITE message. Please refer to lines 4.01-4.03 for invitation, and to lines 5.01-5.03 for join. The intra-overlays in which a joining node will act can be chosen in different ways. A peer receiving an invitation to join a network through the INVITE message sent by another node can evaluate, via the good_deal? social-based primitive, the relevance of this invitation. If the invitation was positively evaluated, it can send a JOIN message to the peer that launched the invitation. Upon receipt of a JOIN message, a peer can decide, again via the good_deal? primitive, whether or not this join is interesting for it.

IV. The Simulations

The purpose of simulations is here to allow to better understand the behavior of platforms interconnecting structured overlay networks through the Synapse approach. We focus on key metrics traditionally considered in distributed information retrieval process, such as exhaustiveness (ratio of existing objects effectively retrieved by the protocol), latency (number of hops to reach the requested object) and the amount of communications produced (number of messages generated for one request). We want to highlight the behavior of these metrics while varying the topology (number of synapses, connectivity of synapses, TTL, number of intra-overlays, etc.).

A. Settings

Our simulations have been conducted using simple Python scripts. The global overlay network considered is a set of Chord networks interconnected by synapses. Information is a set of (key,value) pairs. Each pair is unique and exists once and only once in the network. We study the unstructured interconnection of structured networks. We used discrete-time simulation: queries are launched on the first discrete time step, initiating a set of messages in the network, each message sent at the current step will be received by its destination (next routing hop) at the next hop.

B. Synapses

Our first set of simulations has the intent of studying how the previously mentioned metrics vary while we add synapses or increase their degree (the number of intra-overlays a co-located node belongs to). The number of nodes was fixed to 10000, uniformly distributed amongst 20 overlays. Queries are always triggered by one random node, the key sought by a query is also picked uniformly at random among the set of keys stored by the network. A query is said to be *satisfied* if the pair corresponding to the key has been successfully retrieved.

Figure 4 shows the evolution of the exhaustiveness while increasing the average number of overlays a synapse belongs to. We repeated the experiment for different ratios of synapses (in percentage of the total number of nodes). The exhaustiveness is improved by increasing both factors. We obtain more than 80% of satisfaction with only 5% of nodes belongs to 10 floors, other nodes belonging to only one intra-overlay. When each node belongs to 2 overlays, the exhaustiveness is also almost guaranteed.

We also study search latency, i.e. the number of hops to obtain the first successful response. As illustrated in Figure 5, one first point to notice is that the number of hops remain logarithmic in a Synapse network (number of nodes is 10000, latency never exceeds 14). Other experiments conducted by increasing the number of nodes confirm it. More precisely, Figure 5 highlights the following behavior: (i) when the network contains only few synapses, the latency first increases with the degree of synapses: only few close keys are retrieved. (ii) Then, when both parameters (connectivity and number of synapses) have reached a threshold, the searches can touch more synapses, and the whole network becomes progressively visible, multiple parallel searches become more and more frequent and distant nodes are reached faster. Obviously, multiple searches in parallel lead to an increased number of messages. As illustrated on Figure 6, this number can

[8]Please refer to the Synapse web page for more information

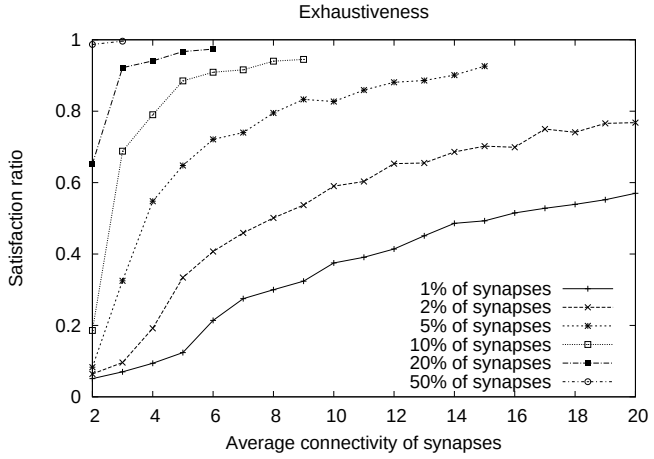

Fig. 4. Synapses and exhaustiveness

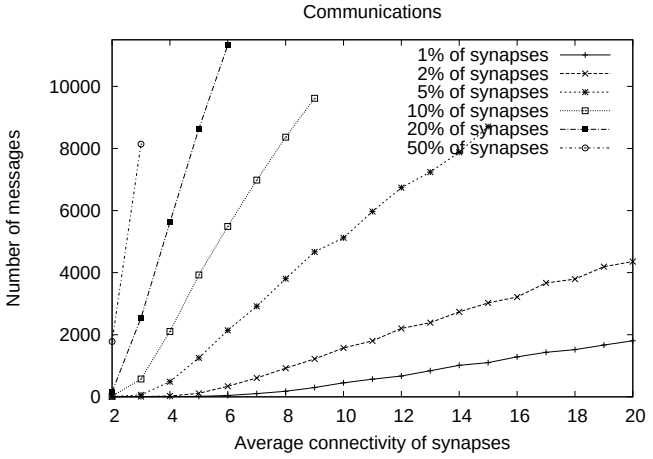

Fig. 6. Synapses and communications

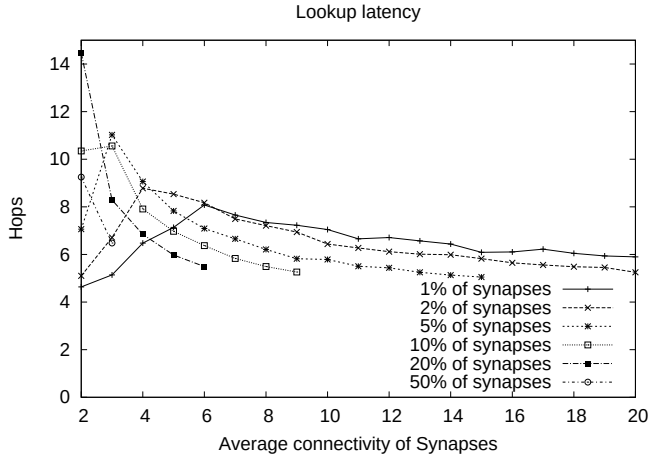

Fig. 5. Synapses and latency

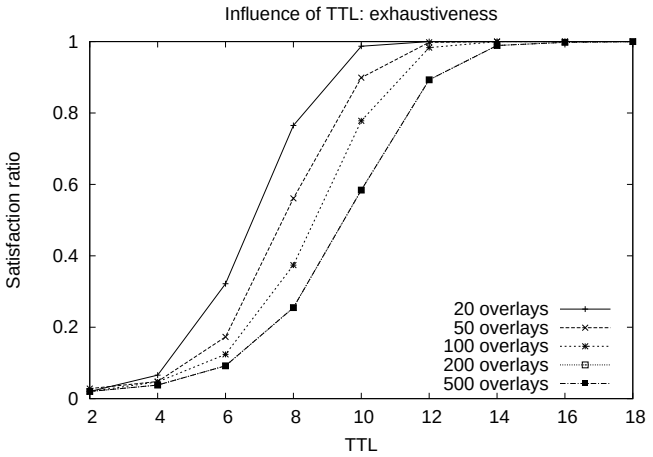

Fig. 7. TTL and exhaustiveness

increases proportionally with the connectivity and number of synapses. A good-deal strategy in synapses can leverage this inter-network overhead.

C. Time-To-Live

We launched a second set of experiments in order to study the impact of the TTL on the search queries. This TTL is simply decreased every time the query traverses a node. The purpose is here to preserve a significant exhaustiveness while reducing the amount of communications undergone by the inter-overlay. We also made the number of overlays vary, to experiment the impact of the *granularity* of the network. In other words, a Synapse network made of few large structured intra-overlays could be called *strongly structured*, and another network with many smaller structured intra-overlays could be called *weakly structured*. The number of nodes is still set to 10000, and every node is a synapse belonging to 2 overlays chosen uniformly at random.

Figure 7 confirms that this low connectivity (2) is enough to achieve quasi-exhaustiveness. Another interesting result is that TTL can be bounded without any impact on exhaustiveness (10 or 12 is enough even when the number of overlays interconnected is 500), while, as highlighted by Figure 8, drastically reducing the amount of communications experienced, the number of messages being almost divided by 2. Finally, as can we can see on both Figures 7 and 8, the *granularity* does not significantly influence exhaustiveness and communications when the number and connectivity of the synapses are fixed.

D. Peers' churn

Since networks are intended to be deployed in dynamic settings (nodes joining and leaving the network without giving notice), we conducted a final set of simulations to see the tolerance of Synapse compared to a single overlay network. Our simulations are again based on Chord. In other words, the question is *Would an interconnection of small Chords better tolerate transient failures than one large Chord?* In this experiment, at each step, a subset of nodes is unreachable, making message routing fail. As we can see on Figure 9, a real improvement on the number of satisfied requests can be obtained through a Synapse like network: When the probability

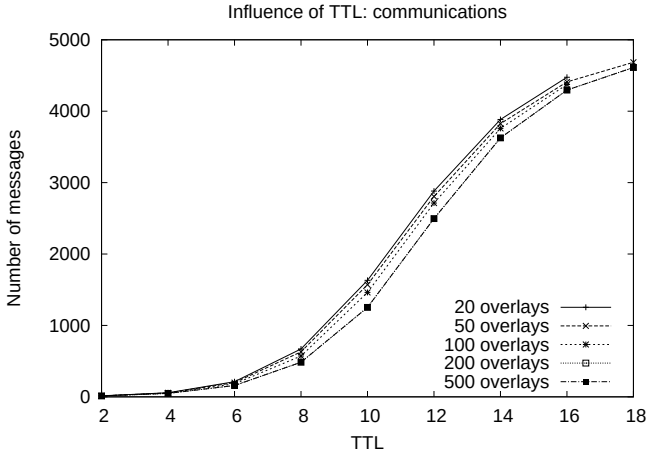

Fig. 8. TTL and communications

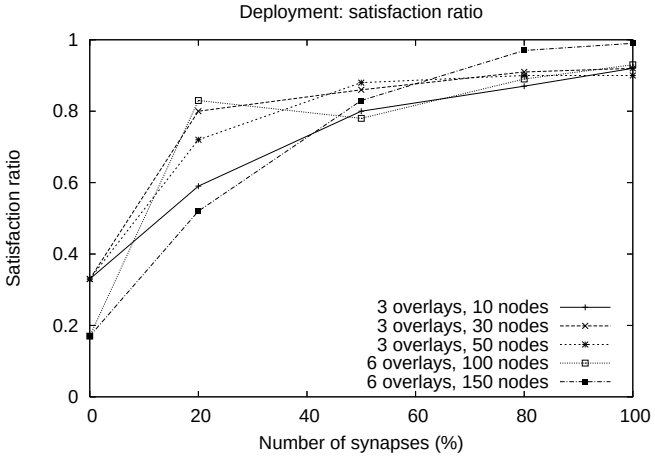

Fig. 10. Deploying Synapse on the `Helios` cluster - Exhaustiveness

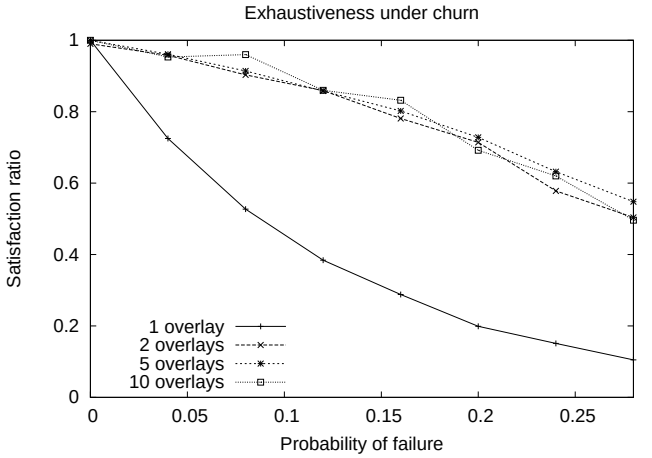

Fig. 9. Churn and exhaustiveness

of failure at each step of a node increases, the global availability of the network is far less reduced, making such architecture more robust on the client's point of view and thus good candidates for information retrieval on dynamic platforms.

V. The Experimentations

A. JSynapse

In order to test our protocols on real platforms, we have first developed JSynapse, a Java-based software prototype, which uses Java RMI standard for communications between nodes, and whose purpose is to capture the very essence of our Synapse protocol. It is a flexible and ready to be plugged library which can inter-connect any type of overlay networks. This prototype fully implements a Chord-based inter-overlay network along with our original Maximum Replication Rate mechanism. It is a lightweight easy to extend software. We also provided some practical classes which help in automating the generation of the inter-overlay network and the testing of specific scenarios.

We have experimented JSynapse on the Grid'5000 platform connecting more than 20 clusters on 9 different sites. Again, Chord was used as the intra-overlay protocol.

In our first deployments, we used one cluster located at Sophia Antipolis, France. The `Helios` cluster is made of 56 quad-core AMD Opteron 275 linked by a gigabit Ethernet connection. The Synapse network created was first made of up to 50 processors uniformly distributed among 3 Chord intra-overlays. Then, still on the same cluster, as nodes are quad-core, we deployed up to 3 logical nodes by processor, thus creating a 150 nodes overlay network, nodes being dispatched uniformly over 6 overlays. During the deployment, overlays were progressively bridged by synapses (whose degree was always 2).

Our goal is here to give a proof of concept and to show the viability of the Synapse approach while confirming results obtained by simulation. We also focused on the metrics affecting the user (satisfaction ratio and time to get a response). In our deployment, once his request was sent, a user waits only 1 second before closing channels opened to receive responses. If no response was received after 1 second, the query is considered as not satisfied.

Figure 10 shows the satisfaction ratio when increasing the number of synapses. As expected, the general behavior is comparable to the simulation results and a quasi-exhaustiveness is achieved, with only a connectivity of 2 for synapses.

Figure 11 illustrates the fact that, the latency experienced by the user when launching a request is very low (few milliseconds) even when a lot of synapses may generate a lot of messages. Obviously this result has to be considered while keeping the performances of the underlying hardware and network used in mind. However, this suggests the viability of our protocols, the confirmation of simulation results, and the efficiency of the software developed.

B. Open-Synapse

We have also developed `open-synapse`, based on the stable and widely used `open-chord` implementation, which

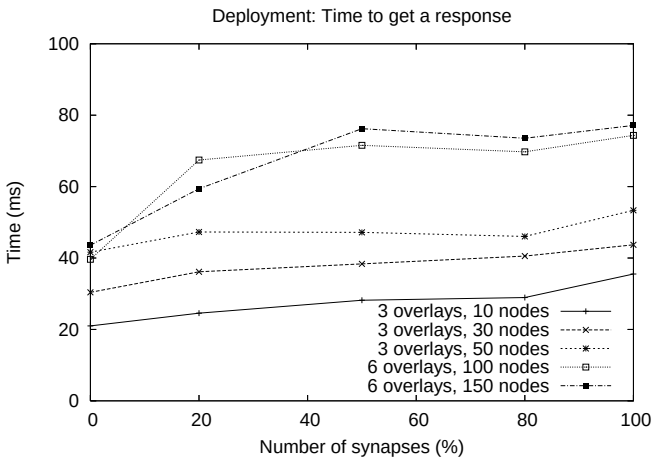

Fig. 11. Deploying Synapse on the `Helios` cluster - Latency

provides a complete and efficient Chord implementation. Open-Synapse has been developed as an extension of the `open-chord` core and thus takes advantage of its robustness and reliability. A preliminary set of tests have been successfully conducted on `open-synapse`, involving 50 nodes and different randomly generated scenarii.

VI. Conclusion

We have introduced Synapse, a scalable protocol for information retrieval in heterogeneous inter-connected overlay networks relying on co-located nodes. Synapse is a generic and flexible *meta*-protocol which provides simple mechanisms and algorithms for easy overlay networks' interconnection. We have given the set of algorithms behind our protocols and provided a set of simulations allowing to capture the behavior of such networks and shows their relevance in the context of information retrieval, using key metrics of distributed information retrieval. We have also developed `JSynapse`, a lightweight implementation of Synapse, and experimented it on the Grid'5000 platform, thus confirming our simulation and giving a proof of concept of our protocol.

As future work, we first intend to focus on social mechanisms involved in synapses, which can greatly influence the shape of the network. On the practical side, we plan to extend `JSynapse` and plug in some other overlay network implementations. More intensive tests and deployments of `open-synapse`, our early prototype based on `open-chord` will also be considered.

Acknowledgment

The authors warmly thank Ernst Biersack for the precious help in reading an early version of this paper, Arnaud Legout and Fabrice Huet for the useful discussions.

References

[1] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Adressable Network," in *ACM SIGCOMM*, 2001.

[2] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup service for Internet Applications." in *ACM SIGCOMM*, 2001, pp. 149–160.

[3] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-To-Peer Systems," in *Int. Conference on Distributed Systems Platforms (Middleware)*, 2001.

[4] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: A Resilient Global-scale Overlay for Service Deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 41–53, 2004.

[5] G. S. Manku, M. Bawa, and P. Raghavan, "Symphony: Distributed Hashing in a Small World," in *USITS'03: Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems*, 2003.

[6] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "SCRIBE: A Large-Scale and Decentralized Application-Level Multicast Infrastructure," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 20, 2002.

[7] L. Liquori, D. Borsetti, C. Casetti, and C. Chiasserini, "An Overlay Architecture for Vehicular Networks," in *IFIP Networking, International Conference on Networking*, ser. LNCS, vol. 4982. Springer, 2008, pp. 60–71.

[8] R. Chand, M. Cosnard, and L. Liquori, "Powerful resource discovery for Arigatoni overlay network," *Future Generation Computer Systems*, vol. 1, no. 21, pp. 31–38, 2008.

[9] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-peer Support for Massively Multiplayer Games," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Society*, vol. 1, 2004, p. 107.

[10] T. Schütt, F. Schintke, and A. Reinefeld, "Scalaris: Reliable Transactional P2P Key/Value Store," in *Proceedings of the 7th ACM SIGPLAN workshop on ERLANG*. ACM, 2008, pp. 41–48.

[11] X. Zhang, J. Liu, B. Li, and Y.-S. Yum, "CoolStreaming/DONet: a Data-Driven Overlay Network for Peer-to-Peer Live Media Streaming," in *INFOCOM 2005.*, vol. 3, 2005, pp. 2102–2111 vol. 3.

[12] X. Shen, H. Yu, J. Buford, and M. Akon, Eds., *Handbook of Peer-to-Peer Networking*. Springer-Verlag, 2010, to appear.

[13] M. Siekkinen, V. Goebel, T. Plagemann, K. Skevik, M. Banfield, and I. Brusic, "Beyond the Future Internet–Requirements of Autonomic Networking Architectures to Address Long Term Future Networking Challenges," in *International Workshop on Future Trends of Distributed Computing Systems*, 2007, pp. 89–98.

[14] A. Fonte, M. Curado, and E. Monteiro, "Interdomain Quality of Service Routing: Setting the Grounds for the Way Ahead," *Annals of Telecommunications*, vol. 63, no. 11, pp. 683–695, 2008.

[15] L. G. Erice, E. W. Biersack, K. W. Ross, P. A. Felber, and G. U. Keller, "Hierarchical P2P Systems," in *Proc. of Euro-Par*, 2003.

[16] Z. Xu, R. Min, and Y. Hu, "HIERAS: A DHT Based Hierarchical P2P Routing Algorithm." in *ICPP*, 2003.

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

[18] T. M. Shafaat, A. Ghodsi, and S. Haridi, "Handling network partitions and mergers in structured overlay networks," in *Proc. of P2P*. IEEE Computer Society, 2007, pp. 132–139.

[19] D. Borsetti, C. Casetti, C. Chiasserini, and L. Liquori, *Heterogeneous Wireless Access Networks: Architectures and Protocols*. Sringer-Verlag, 2009, ch. Content Discovery in Heterogeneous Mobile Networks.

[20] L. Cheng, R. Ocampo, K. Jean, A. Galis, C. Simon, R. Szabo, P. Kersch, and R. Giaffreda, "Towards Distributed Hash Tables (De) Composition in Ambient Networks," *LNCS*, vol. 4269, p. 258, 2006.

[21] L. Cheng, "Bridging Distributed Hash Tables in Wireless Ad-Hoc Networks," in *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, 2007, pp. 5159–5163.

[22] M. Kwon and S. Fahmy, "Synergy: an Overlay Internetworking Architecture," in *Proc. of International Conference on Computer Communications and Networks*, 2005, pp. 401–406.

[23] K. Junjiro, W. Naoki, and M. Masayuki, "Design and Evaluation of a Cooperative Mechanism for Pure P2P File-Sharing Networks," *IEICE Trans Commun (Inst Electron Inf Commun Eng)*, vol. E89-B, no. 9, pp. 2319–2326, 2006.

[24] P. Furtado, "Multiple Dynamic Overlay Communities and Inter-space Routing," *Lecture Notes in Computer Science*, vol. 4125, p. 38, 2007.

[25] L. Liquori, C. Tedeschi, and F. Bongiovanni, "BabelChord: a Social Tower of DHT-Based Overlay Networks," in *14th Symposium on Computers and Communications (ISCC 2009)*. IEEE, 2009, short paper.