Efficient Cooperative Discovery of Service Compositions in Unstructured P2P Networks

Angelo Furno, Eugenio Zimeo
University of Sannio
Department of Engineering
82100, Benevento - Italy
{*angelo.furno, eugenio.zimeo*}@*unisannio.it*

***Abstract*—In this paper, we propose an efficient technique for improving the performance of automatic and cooperative compositions in P2P unstructured networks during service discovery. Since the adoption of flooding to exchange queries and partial solutions among the peers of unstructured networks generates a huge amount of messages, the technique exploits a probabilistic forwarding algorithm that uses different sources of knowledge, such as network density and service grouping, to reduce the amount of messages exchanged. The technique, analyzed in several network configurations by using a simulator to observe resolution time, recall and message overhead, has shown good performances especially in dense and large-scale service networks.**

***Keywords*-Service Discovery; Semantic Overlay Networks; Service Composition; Semantic Web Services; Peer-to-Peer Computing.**

I. Introduction

Service discovery enables consumers to find, in a growing space of available services, the desired ones. It is highly likely that, as the number of services to handle grows, more scalable architectures than centralized ones will be needed to implement service registries and their discovery capabilities. One solution could be the adoption of decentralized approaches based on a hierarchical architecture similar to the one adopted by the Domain Name System (DNS).

Since services can be linked together to create solutions for complex needs, service discovery becomes a very complex process if compared with DNS resolution queries: when the desired services are not available in isolation, solutions can still be found by combining multiple services to form compositions. This way, discovery can exhibit a higher level of recall (i.e. the number of solutions found with respect to the number of solutions actually present in the network) compared to the well-known discovery of atomic services.

Peer to peer (P2P) architectural models are the best candidates to implement future-generation service registries and discovery mechanisms. These models, in fact, ensure high functional and non-functional scalability: **(1)** by using a P2P registry on a large-scale network, like the Internet, it is possible to access to a very large distributed repository of services, belonging to different organizations; **(2)** large repositories can be explored through several discovery processes running in parallel over the network peers. Moreover, P2P service registries enable a new form of collaboration where the roles of consumers and providers can be used interchangeably to cooperatively create service compositions that satisfy consumers' queries.

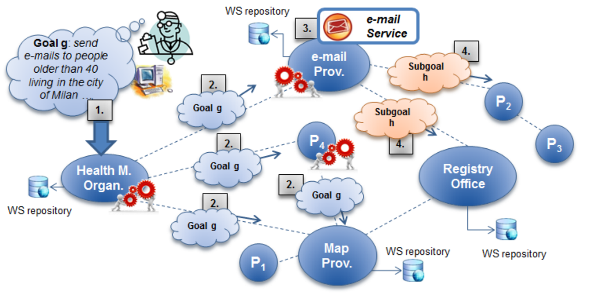


Figure 1. Collaborative composition process for the example scenario

As an example (see Fig. 1), a health maintenance organization could need to inform people suffering from some illness or at-risk patients to go to a hospital for a check-up within a specified date. The check-up notification has to be delivered to people living in a specific geographical area, within the range of the health organization, and with some specific age requirement: e.g. alerting people living within a range of 5 kilometers, older than 40 years or suffering from heart related illnesses, to contact the nearest hospital and make an appointment for a cardiovascular check-up.

Fig. 1 shows a P2P service network, including peers with useful services for solving the complex goal (e.g. *e-mail service provider*, etc.), non-relevant peers (P_1, P_2, ...) and the communication links (dashed lines). Also, by means of numbered labels, a few stages related to the execution of a possible P2P collaborative composition process are shown. The gear symbols are used to point out the peers actually involved in the stages and collaborating to find composite solutions. Fig. 2 presents the resulting composite service.

In this paper, by exploiting our previous experience on service composition [1]–[3], we propose an efficient technique for service composition in unstructured P2P service networks that is able to take into account several informa-

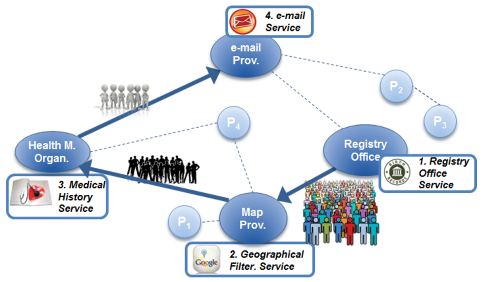


Figure 2. The resulting composite service and its distributed execution

tion about the networks to strongly reduce the number of messages exchanged, if compared with optimized flooding.

Comparison with other techniques is only possible from a conceptual point of view, since very few approaches address service composition over P2P unstructured networks and, in most of the cases, the code is not publicly available.

In our model, any peer of a service network can publish semantically described services in a local registry and perform local or distributed discovery of both atomic and composite services, whose parts could be allocated to any of the peers' registry. The network dynamically evolves by changing its virtual topology according to the links among the services participating to compositions. This knowledge, together with information about the connectivity graph (e.g. network density), drives query propagation in a more and more precise and dynamic way, as long as the network evolves during its use. This way, a service network can change from the initial completely unstructured organization to a semi-structured organization in the steady state phase.

The remaining part of the paper is organized as follows. Section II presents the main research efforts related to this work; Section III introduces the concept of distributed and cooperative composition and the algorithms used for P2P discovery and composition; Section IV presents the solution used to create a self-evolving semantic service network; Section V describes the probabilistic strategy adopted for avoiding flooding in P2P unstructured networks when forwarding service requests; in Section VI, simulations and results are discussed; finally, Section VII concludes the paper and highlights future work.

II. Related Work

Service discovery often relies on the use of centralized registries, discovery engines or brokers. To improve scalability, dynamicity and robustness, in recent years, some researchers [4]–[6] have exploited DHT-based networks (e.g. Chord [7]). They are relatively simple to handle during discovery but present some drawbacks: **(a)** high churn overhead, especially in networks where peers and services appear and disappear frequently (e.g. Chord churn complexity is $O(log_2(N))$ [7]); **(b)** services are strongly tied to provider peers; **(c)** hash functions of DHT-based networks make it easy ID-based exact matching between queries and contents, but semantic matching needs complex hash functions.

Consequently, unstructured P2P networks are acquiring growing consensus [1], [2], [8]–[10] for supporting semantic service discovery, due to their flexibility, fault tolerance and semantic matching capabilities: any peer can publish its services in a local repository and semantic queries of the desired service may be effectively routed towards the right registry. However, very few works address the problem of efficient service composition in unstructured P2P networks, since most proposed solutions for both service discovery and composition are based on flooding (e.g. Gnutella [11]), which generates an important message overhead, causing high routing costs and low scalability [12].

Semantic overlays have been proposed in [1], [2] to improve efficiency of P2P discovery of service compositions. The authors foster the creation of groups of peers, hosting the services previously used to create compositions. The groups are managed by superpeers, who are in charge of addressing the service requests to the groups that host compositions, so increasing the probability of finding the desired service in a shorter time. Service composition is performed by exploiting traditional approaches typically applied to centralized repositories [13]: they adopt AI planning techniques, and semantics to improve automation capabilities.

An approach on semantic P2P overlay networks with superpeers is also presented in [9]. However, their JXTA [14] based discovery framework performs only simple service discovery, with no possibility to retrieve composite solutions. Also, query propagation is performed by means of simple flooding inside the groups managed by superpeers.

In [10], the authors propose to use a caching mechanism to increase efficiency in solving service queries through service composition. The approach has some similarities with our solution, but differently, when no entry is present in the composition cache, the resolution strategy still relies on flooding, generating a relevant message overhead. Also, no strategy for propagating service requests towards the most promising directions (semantic routing) is provided to lower resolution time, as allowed in our approach by exploiting multiple semantic overlays managed by superpeers.

III. Service Discovery and Cooperative Composition Work

In the following, we refer to abstract services and goals: they are both semantically described, through the use of OWL ontologies and OWL-S descriptions. A service, and similarly a service goal, is described as an OWL-S profile, whose precondition and effect elements represent pre- and post- conditions of the service and are described by using SWRL expressions referring to OWL ontologies. More details on these aspects can be found in our previous work [3], where a centralized approach for service composition and an associated tool for automatically generating executable WS-BPEL processes are described.

Since in this paper we are mainly interested in the distributed aspects of composition, pre- and post-conditions will be represented, for the sake of simplicity, with tokens. Semantics-based representation of pre- and post-conditions could further support query resolution by improving recall and/or precision during search.

Each peer of the network is involved in the concurrent execution of four main protocols: **(1)** discovery of services satisfying user goals, allowing for collaborative and distributed composition; **(2)** query forwarding for efficiently propagating service requests in the network; **(3)** network reorganization for building semantic service overlays from already solved service requests; **(4)** gossiping, as part of the forwarding strategy, for disseminating knowledge about the network structure among the peers. The discovery protocol is described in this section, network reorganization in Section IV, forwarding and gossiping in Section V.

The protocol executed by each peer to perform P2P service discovery or composition is organized in two threads and described in pseudo-code in Listings 1 and 2. The spawn keyword indicates asynchronous invocation of the associated procedure (i.e. a new thread is spawned for the procedure).

```
1  discoveryProtocol(Goal g) [active thread]
2    Query q := createQuery(g);
3    spawn forward(q);
4    spawn solve(q);
5    spawn reorganizeNetwork(g);
```

Listing 1. Discovery Protocol, active thread

```
1  discoveryProtocol() [passive thread]
2    do forever
3      receive(q, sols);
4      if sols is nil  /*case 1: handle new goal request */
5        if q.TTL <= 0 || q has already been solved
6          continue;
7        else
8          spawn forward(q);
9          spawn solve(q);
10     else        //case 2: handle solutions for query q
11       if q.refQuery is not nil /*case 2a: merge partial
              solutions*/
12         localPartSols := partialSolsTable[q.refQuery];
13         newSols := merge(q.refQuery, sols, localPartSols);
14         reply (q.refQuery, newSols) to q.refQuery.source;
15       else //case 2b: notify complete solutions
16         add sols to completeSolsTable[q];
17         locally notify sols for q.goal;
```

Listing 2. Discovery Protocol, passive thread

The *active thread* generates a new query object from the user specified goal (`createQuery`, line `2`), by setting information like the query's Time To Live (TTL), a query identifier, the query goal and the source peer identifier. The query is then forwarded to other peers according to the `forward` procedure (line `3`). Since we adopt a top-down approach to present the algorithms, the `reorganizeNetwork` and `forward` procedures, implementing our self-evolving overlay network strategy and probabilistic forwarding technique, will be described in Sections IV and V, respectively. After the query has been forwarded, the solving procedure (`solve` algorithm in Listing 3) is started for the newly generated query.

The procedure `localSearch` in line `2` of Listing 3, is based on a semantic matching process (not shown in the paper because outside its scope) and explores the local peer repository in order to find complete solutions to the requested goal. To this end a semantic matching step between the goal and the locally available service descriptions is performed. The matchmaking algorithms, used in the search phase, are extensively described in our previous works [15], [16], but other algorithms or tools could be used [17], [18].

When no complete solution is available in the peer repository, partial solutions to the goal query are searched, by using a backward strategy (line `6`). The `localBackwardSearch` is essentially a variant of the `localSearch` one: a partial solution to the submitted goal g matches its post-conditions, while having different pre-conditions. The same matchmaking algorithms are used to decide the presence of a match on post-conditions.

```
1  solve(Query q)
2    localSols := localSearch(q, localRepo);
3    if localSols is not empty
4      reply(q, localSols) to q.source;
5    else
6      localPartialSols := localBackwardSearch(q, localRepo);
7      add localPartialSols to partialSolsTable[q];
8      foreach partialSol in localPartialSols
9        partialGoalQuery:= createGapQuery(q.goal, partialSol);
10       spawn forward(partialGoalQuery);
11       spawn solve(partialGoalQuery);
```

Listing 3. Query solving procedure

Whenever a partial solution is found, a new query is created for the goal ranging from the pre-conditions in the original goal to the pre-conditions of the found partial solution (line `9`). The newly generated query is forwarded to other peers according to the `forward` procedure (line `10`). Also, the `solve` procedure is called recursively (line `11`) to find other complete or partial solutions to the new query on the peer that generated it. Partial solutions, when received, have to be merged to previously found local solutions related to the same partial query (passive thread, lines `11÷14`) and complete solutions have to be sent back to the query source peer. The submitter receives the service composition (i.e. the list of the composing services and the peers hosting them).

The `reorganizeNetwork` procedure is spawned (line `5`) in the active thread to start the network reorganization phase. It is a concurrent process that waits for query solutions coming from the network before starting the network reorganization according to the inferred knowledge.

The *passive thread* in Listing 2 is meant to awake a peer when: **(1)** a new query has to be solved (lines `5÷9`); **(2)** new solutions are available (lines `11÷17`).

In the first case **(1)**, the peer first starts the `forward` procedure (line `8`) to effectively and efficiently spread the query through the network, and then starts the `solve` procedure (line `9`) in order to find solutions.

In the second case **(2)**, the peer verifies whether the received solutions refer to a partial or a complete goal and decides the next operations to perform. If the query refers to a partial goal (lines `11÷14`), partial solutions to the referred query are retrieved from the `partialSolsTable` and merged (`merge` procedure, line `13`) with the newly received solutions to create a new composite solution, which will be a complete solution for the referred query. The new merged solutions are sent back to the peer who has generated the referred query (line `14`). Instead, if the solved query refers to a complete goal (i.e. referred query is `nil`, line `15`), the peer has just received complete solutions to a local generated goal request and solutions are stored locally (`completeSolsTable`), to be used in the network reorganization phase (Section IV), and notified to the requesting user (line `17`).

IV. Self-evolving Semantic Overlays

To model the collaborative backward strategy above, we adopted a P2P superpeer architecture, a special kind of unstructured P2P networks. Some special peers, called superpeers, act like proxies, forwarding queries to groups of managed simple peers.

Each simple node of the network runs the described protocols, enabling publishing (which presumes a local repository), discovery and composition mechanisms on it. Superpeers are needed when new acquired service knowledge to manage exists: solutions are found that can be useful for successive queries. Each cooperative group is handled by a superpeer, controlling the cooperating peers, by managing links to them, storing a description of the group and properly forwarding queries to the peers in the group when needed.

A node is a computer, a virtual machine and any other software/hardware device that is able to execute the discovery protocol described in Listings 1 and 2. On the same node, there can be, in general, multiple peer processes, running the proposed search and self-evolution algorithms. Some of them may act as superpeers. At the node abstraction layer, we assume the existence of a connectivity graph, representing the links initially known to the nodes of the network. Graph links can define any complex network topology (a mesh, a ring, a tree, etc.) and can be associated to physical proximity or low latency paths among the network nodes.

The `reorganizeNetwork` procedure, introduced in Section III, is described in Listing 4.

```
1  reorganizeNetwork(Query q)
2    wait(networkReorganizationTimeout);
3    solutions := completeSolsTable[q]
4    peersSet := retrieveParticipants(solutions);
5    if peersSet.size <= 1
6      return;
7    foundGroups := new Set<Group>();
8    foreach peer in peersSet
9      peerGroups := retrieveGroups(peer, q.goal);
10     if peerGroups is not empty
11       remove peer from peersSet;
12       add peerGroups to foundGroups;
13   intersections := retrieveIntersections(foundGroups);
14   reorganizeGroups(foundGroups, intersections);
15   if peersSet is not empty
16     createNewGroup(peersSet, solutions);
```

Listing 4. Network Reorganization Protocol

Starting from the initial connectivity graph, when a composite solution is identified, the network implicitly aggregates the participating peers to form a new group. A timeout mechanism is used for stopping the reception of solutions for the locally generated query and starting network reorganization. The process is executed continually in the network, giving rise to several virtual layers of links (service composition overlay network), placed over the connectivity graph and managed at the superpeer level. Superpeers maintain both pointers (peer ID) to the peers making up the new composite solution and a semantic characterization of the solved goal. Simple peers maintain pointers to their superpeers and a semantic characterization of the groups they belong to. The links of the semantic overlay networks, stored at the superpeer level, make it possible the routing of new queries along the most convenient direction for resolution. The self-evolution algorithm is started and orchestrated by the peer who submitted a service request (line `5` in Listing 1).

Also, in order to maximize reuse of previously acquired knowledge, we reorganize overlapping groups by finding their intersections and storing them into a proper collection of tree-structured data (the intersections variable in line `13`). To this purpose the `retrieveIntersections` procedure is used. Any intersection found is turned into a new group with its semantic description and superpeer. The superpeer of any intersection group has to appear as member in any other group owning that intersection. This way, a hierarchical structure is given to the overlay.

When new groups are created or existing ones are modified, new found composite solutions are transferred from the submitter to the group superpeer, which publishes them on its local repository. The repository will be explored by the superpeer when new queries are received, allowing the reuse of already performed service compositions. This way, it is possible to more quickly solve queries which are semantically similar or identical to previously solved ones, without generating another distributed backward search (even if propagation in the group would be performed efficiently by exploiting group semantic links).

V. Query Selective Forwarding

In a P2P network, avoiding flooding is essential to reduce the number of messages exchanged for solving discovery query and to ensure, at the same time, lower composition times without significantly affecting the ability of finding all the existing solutions in the network. To this end, we propose a novel probabilistic forwarding algorithm that exploits knowledge about both the connectivity network (such as network density) and the virtual overlay networks, created to host already found compositions and reduce the number of messages exchanged in the network.

```
forward(Query q)
  if isSuperpeer(this)
    foreach group[matching q.goal] in this.groups
      reply(q, nil) to group.superpeer;
  else
    propagateQueryToNeighbors(q);
    if (isNeighbor(q.sender) and not isSuperpeer(q.sender)) or
           query submitted locally
      foreach group[matching q.goal] in this.groups
        reply(q, nil) to group.superpeer;
```

Listing 5. Query forwarding procedure

The overall forwarding algorithm (Listing 5) distinguishes two cases: **(1)** Superpeers forward queries to the other known superpeers, managing semantically goal-related groups (lines `3, 4`). This is done to facilitate query propagation to groups having more useful information for query resolution; when such groups do not exist, superpeer propagation can be completed by forwarding the query to neighbors over the connectivity graph, during the execution of the `superpeerSolve` procedure; **(2)** Simple peers forward queries to their neighbor nodes on the connectivity graph (line `6`) and to the superpeers of the groups they belong to (line `9`). Specifically, only those groups semantically related to the goal in the query are considered, in the case the request has been originated locally to the peer or has been received from a neighbor simple peer (lines `7, 8`).

The algorithms `superpeerSolve` (Subsection V-A) and `propagateQueryToNeighbors` (Subsection V-B) complete the global forwarding procedure.

A. Superpeer Propagation Algorithm

The `superpeerSolve` algorithm is the search procedure executed by superpeers, in place of the `solve` one in Listing 3, executed by simple peers. The only differences with `solve` are: **(1)** searching is performed over the group solutions stored in the superpeer repository (i.e., previously found composite solutions); **(2)** the algorithm includes propagation of the received query to the superpeer node's neighbors, when no complete solution is found.

As regarding point **(2)**, the propagation over the connectivity graph is not performed immediately upon receiving the query, as in the case of simple peers, but only when it has been verified that there is no complete solution on the superpeer. This avoids message overhead when a complete solution can be directly found on the superpeer, by exploiting the larger knowledge available at this level. This consideration has to be related to the main forwarding algorithm in Listing 5, where the `else` body (lines `6`$\div$`9`, related to the simple peer case) contains the propagation to neighbors at line `6`, while the `if` body (lines `3, 4` for the superpeer case) contains none.

B. Propagation over the Connectivity Graph

The algorithm `propagateQueryToNeighbors` implements the forwarding mechanism adopted by every peer on the nodes of the connectivity graph and represents the core of the probabilistic forwarding mechanism.

In a traditional flooding approach, every message received from a peer is forwarded to all the neighbors of the peer. Even if most of the works addressing service composition in unstructured P2P networks uses flooding for propagating messages, we also consider an optimized variant of this basic flooding approach, that we call *optimized flooding* in the rest of the paper, as a technique to compare our probabilistic approach. With optimized flooding, the query message to propagate maintains the list of local neighbors the query has already been forwarded to by the node from which the query has been received. When another node has to decide about propagation, the list is considered to exclude local neighbors that already received the query from the sender. The list of neighbors in the propagated query is then updated with the new information about the actual new forwarded nodes.

Our probabilistic forwarding algorithm is based on the introduction of a propagation threshold, τ in the following, limiting the number of neighbors to which the query should be forwarded (*forwarded peers*). The propagation threshold represents the fraction of neighbors to select for propagation and is dynamically computed anytime the algorithm is executed, using network information. The number of forwarded peers (f) is computed according to the formula:

$$f = \lceil \tau \cdot \lambda \rceil, \text{ where: } \lambda = \begin{cases} l-1, & \text{if sender is a neighbor} \\ l, & \text{otherwise} \end{cases},$$

being l the neighborhood size for the current node. f represents the maximum number of neighbors to contact for query propagation. To select up to f neighbors to forward the query to, the propagation algorithm takes into account the information related to the previous forwarded nodes, like in the optimized flooding approach, coded in the received query message by the sender. By excluding the possible common neighbors that has already received the query from the sender, up to f neighbors are randomly selected among the whole neighborhood and stored as the new list of propagated nodes in the query message. Finally the message is forwarded to them.

We have identified three different kinds of network information, which can be computed at any time by a peer for the dynamic evaluation of τ: **(1)** Availability of relevant service composition overlays; **(2)** Global density of the network; **(3)** Number of peers (hops) crossed by the goal query from the source to the current peer. The information above makes it possible to distinguish three contributions to threshold τ, which we denote as τ_{Groups}, $\tau_{Density}$ and τ_{Hops}.

The value of threshold τ is evaluated as a weighted (the weights ω_{Groups}, $\omega_{Density}$ and ω_{Hops} are defined at configuration time) and normalized (weights sum up to 1) sum of the three contributions. Each contribution is a threshold itself, is defined in the range $[0, 1]$ and is dynamically computed when the `propagateQueryToNeighbors` algorithm is executed by a peer, using the currently available data. Additional thresholds could be considered if some other knowledge may be accessed in the service P2P networks and used to improve forwarding, by means of a proper strategy and a specific tuning process. We considered the most relevant properties of service networks and kept low the number of thresholds to reduce the overhead tied to the network knowledge retrieval.

In Section VI, we discuss simulations using different weights for the thresholds above. In the following, we explain the semantics of the thresholds and how they are computed by the system to guide to selecting their weights.

1) Semantic Group Threshold (τ_{Groups}): τ_{Groups} is computed considering the number of groups participated and/or controlled by the forwarding peer. Each of these groups is related to previously found service compositions and has a semantic characterization, which makes possible to filter the ones not relevant to the goal resolution. The number of relevant and reachable groups is denoted as η and represents the independent variable of the τ_{Groups} evaluating function.

The rationale behind the evaluation of this threshold is simple: the more semantic links are available for query forwarding, the less it is needed to propagate to neighbors on the connectivity graph. Propagation can be mainly handled through superpeers' connectivity information. Therefore, we identified the following requirements for the evaluating function: if there is no useful group ($\eta = 0$), τ_{Groups} has to be 1, since the query forwarding cannot rely on the semantic forwarding; τ_{Groups} has to be decreasing with respect to η; variable η has theoretically no upper bound, hence τ_{Groups} has to asymptotically decrease to 0 when η increases in the range $[0, +\infty[$. By taking into account these requirements, we have considered the equilateral hyperbola function reported below, which has been tested, in terms of performance indexes, in the simulations of Section VI:

$$\tau_{Groups} = 1/(1 + \eta).$$

2) Density Threshold ($\tau_{Density}$): $\tau_{Density}$ is computed by means of a gossip protocol, continuously performed by every node of the P2P network in parallel with the composition/discovery protocol (Listings 1 and 2). An anti-entropy protocol, based on the push-pull strategy described in [19], has been defined in order to make any node compute the average number of neighbors on the connectivity graph: **(1)** each peer p stores a local approximation of the average number of neighbors in the network as its $state_p$. The initial value is chosen as the number of node's neighbors on the connectivity graph; **(2)** each peer p performs a random selection of the neighbor q to gossip with from its neighborhood; **(3)** when receiving the gossip information $state_p$ from neighbor p, peer q updates its state to the value: $(state_p + state_q)/2$. This state update has been proved to converge to the global state average in [19]. The period for starting a new gossip iteration has to be quite low with respect to the average goal query period, in order to make gossip quickly converge to reliable density information when the discovery/composition process is inactive.

The active thread of the gossip protocol has been designed to work during the inactive phases of our system, i.e. when there are no active goal queries on the peer running over the network nodes, in order not to introduce message overhead during the stages of P2P discovery. However, since the gossip message elaboration overhead is typically low, network density information are also included by each peer within the messages regularly exchanged during the query forwarding messages. This way, it is still possible to update the node's local density information, even when the composition/discovery protocol is active, without generating specific gossip requests.

Finally, in order to know how much the current state is a reliable representation of the global network density, we incrementally compute, on each node, the standard deviation (σ) of the local density information (δ) at any state update. At the beginning of the gossip protocol, we assume an infinite standard deviation. Hence, we introduce a parameter ($\bar{\delta}$), defined as:

$$\bar{\delta} = \begin{cases} \delta - \sigma, & \text{if } \sigma < \delta \\ 1, & \text{otherwise} \end{cases},$$

which we use to compute the $\tau_{Density}$ threshold. The rationale is that the denser is the network (i.e. higher $\bar{\delta}$), the less propagations will be necessary for the query to reach the various peers in the network, because of the presence of many alternative paths. To allow a finer grained control of the density threshold, we have introduced the following parameterization of $\tau_{Density}$:

$$\tau_{Density} = K/(\bar{\delta} + K - 1),$$

with $K \geq 1$. By increasing the K coefficient, it is possible to use a smoother hyperbole in order to softly reduce the number of messages as $\bar{\delta}$ increases. This can be useful to manage message reduction when the network is sparse and a proper number of messages has to be injected to increase the probability of finding solutions.

3) Traversed Hops Threshold (τ_{Hops}): τ_{Hops} is computed by considering the number of hops that a query has already crossed. The idea is that the more hops the query has already crossed the less it is likely a new forward will help in solving it. Hence, we have introduced the independent variable ρ, representing the number of traversed hops. This variable is set to 1 when the query for a goal is first created on a peer, its value is transported by the query itself and is increased by one anytime the query is forwarded from a peer to another one. By using ρ, we have defined the following evaluating function for τ_{Hops}:

$$\tau_{Hops} = 1/\rho.$$

VI. Evaluation

To evaluate the techniques and the algorithms discussed in the previous chapters, we decided to exploit a simulator for implementing the P2P service network and hosting the algorithms. PeerSim [20] is a Java open-source P2P network simulator, including an event-based engine, completely driven by events and node messages, offering concurrency simulation and a simulated transport layer. Several configurations have been considered, in order to evaluate functional correctness and performances of the proposed algorithms. We have focused our attention on three performance indexes:

- *Recall*: system's ability to find simple or complex solutions (among the existing ones) to a goal request, when services published on peers' repositories make it possible to satisfy it. In the following experiments, the recall information is computed as the ratio of the number of solutions found by our system during the simulation cycles and the number of existing ones. In most of the following simulations, only one solution is present in the network at each simulation cycle. Therefore, recall is evaluated with respect to the number of simulation cycles;

- *Message overhead*: the number of request, response and gossip messages exchanged among the peers in the network to find solutions to a goal request. Messages are counted considering the whole network until the first solution to the query under test is received;
- *Resolution time*: the time required for the submitter to receive the first solution to the requested goal.

The P2P network is initialized with a specific number of nodes and a connectivity graph. At the beginning of any simulation, each node of the simulated network hosts one single peer process, able to publish, discover and compose services; no overlay network is available. Service descriptions are published on each peer in the simulation initialization phase or cyclically when we want to evaluate system behavior with respect to different service allocations on the peers, by shuffling service descriptions on available peers. To make simulations consistent with realistic usage scenarios, we introduced three kinds of delay:

- *Semantic elaboration delay*: models the delay introduced by a realistic semantic matching between the query received by the peer and a service description available on that peer. For backward partial resolution, we consider half this delay, since only post-conditions are compared;
- *Transmission delay (t_d)*: models the delay for placing a message from the application layer (peer sending a message) on the network abstraction layer (the node), when no multicast communication is available. If a peer on node A has to send at time t_0 a message to peers on nodes B, C and D, the message for B will be sent at t_0, for C at time $t_0 + t_d$ and for D at time $t_0 + 2 \cdot t_d$;
- *Network latency*: is the latency delay for the simulated transport protocol, used for message exchanges among the network nodes. It represents the time required for a message sent from a node to reach the destination one.

We simulated our P2P network using the following values for the parameters above:

- *Semantic elaboration delay*: $400ms$ for local complete solution; 200 ms for partial backward solution. These delays have been computed by executing a number of queries towards some of the matchmakers used in the S3 contest, the annual contest on Semantic Service Selection [17]. In particular, we focused on ISeM [18], since it offers matching capabilities based on IOPE descriptions;
- *Transmission delay*: $1ms$;
- *Network latency*: uniform random variable in the range $[10ms, 130ms]$. This range refers to latency measured on the Internet when sending small/medium messages to very distant destinations ($130ms$) or very close ($10ms$) ones.

The technique has been evaluated with different sizes of the network (and of published services). For the connectivity graph, we used several topologies, like a star and random meshes, focusing our attention on a dense topology of the network **(1)** and a sparse one **(2)**. In the first case, we used a random mesh where each node has at least $networksize/10$ neighbors (the network become denser as the size increases); in the second one, we used a random mesh with a fixed average number of neighbors (between 2 and 4 neighbors).

As a reference for the reader, we list here the configurations adopted to set weights and coefficients of the probabilistic forwarding algorithm exploited in our experiments. K is the coefficient used in the definition of the hyperbolic function, as described in Section V-B2. Other configurations have been tested in our experiments (e.g. weights for testing thresholds in isolation), but, due to space limitations, only the ones most relevant for the evaluation are reported.

Conf. 1:	$\{\omega_{Groups} = 0.4, \omega_{Density} = 0.3, \omega_{Hops} = 0.3, K = 1\}$
Conf. 2:	$\{\omega_{Groups} = 0.4, \omega_{Density} = 0.3, \omega_{Hops} = 0.3, K = 2\}$
Conf. 3:	$\{\omega_{Groups} = 0.4, \omega_{Density} = 0.5, \omega_{Hops} = 0.1, K = 1\}$
Conf. 4:	$\{\omega_{Groups} = 0.4, \omega_{Density} = 0.5, \omega_{Hops} = 0.1, K = 2\}$

A. *Tested Scenario*

One peer submits a query for a specific goal; each peer hosts one service in its repository and there is only one composite solution in the network, specifically a chain of 10 services published on different peers of the network.

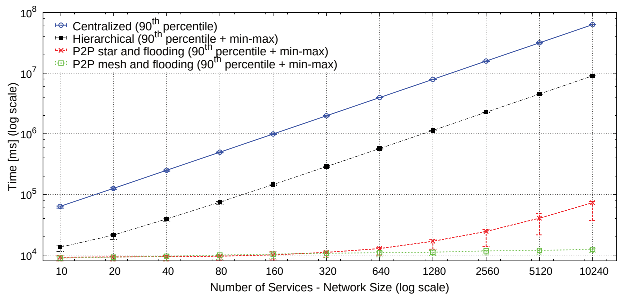
This scenario has been used firstly to the purpose of comparing P2P configurations of the service repository with centralized and hierarchical ones, in order to evaluate the potential benefits of the P2P approach over them (Fig. 3).

In the centralized configuration, one node contains all the services available in the network and discovers or composes services to satisfy requests coming from the other nodes, acting like a public registry in the network.

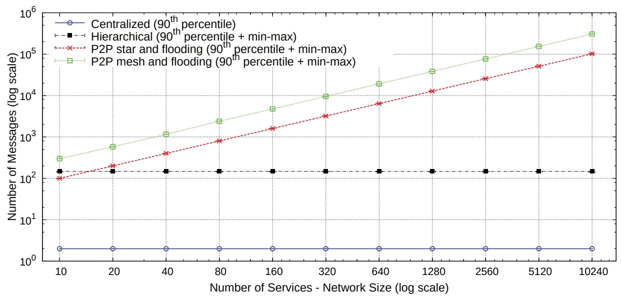
In the hierarchical configuration, services are distributed on a limited (with respect to the network size) number of nodes (7 in our simulations); registry nodes are interconnected according to a hierarchical topology and queries are generated from one node connected as a leaf to the hierarchy.

For the P2P configuration, together with the mesh topology **(2)**, described before, we have also considered a star topology of the connectivity graph, where one single node has links to all the other nodes in the network, as in the centralized configuration, while services are distributed among peers. The simulated delays described at the beginning of this section have been used.

In order to perform a meaningful comparison, the same composition algorithm has been used in each of the above configurations (backward resolution). Also, in order to stress benefits and costs simply coming from the use of our cooperative P2P approach in composition, network reorganization



(a) 10-service composition, resolution time (no semantic reorganization)



(b) 10-service composition, messages (no semantic reorganization)

Figure 3. Resolution times (left) and number of exchanged messages (right) - comparison of repository architectures

has been disabled and traditional flooding has been adopted for query propagation instead of the probabilistic forwarding approach described in Section V.

100 cycles of simulation have been executed. In each cycle, the same query has been issued by one submitter and statistical data acquired. Computer performance indexes are presented as 90^{th} percentiles over the different values collected at the end of each simulation cycle, together with min-max confidence intervals.

Fig. 3a compares the considered configurations, showing the remarkable improvement that can be obtained, in terms of resolution time, by increasing the degree of distribution of the service repository: the centralized configuration presents the highest values; an improvement can be observed when using a hierarchical registry; P2P configurations offer the best resolution time. It is worth to note that the mesh topology exhibits lower values of resolution time, since there is no bottleneck as in the case of the star configuration. Obviously, the more distribution is introduced in the composition process, the higher is the number of messages to be exchanged to find a solution (Fig. 3b).

To evaluate the impact of our probabilistic forwarding technique, the P2P backward composition strategy has been used in conjunction with the probabilistic forwarding algorithm and network reorganization has been activated in order to build overlay networks.

Simulations were performed again in multiple cycles, with one peer issuing a query for the same goal at each cycle. Every two cycles, services were shuffled on the peers. In computing our performance indexes, we have distinguished the average computed over data collected in odd cycles (indicated as 1^{st} query in the following figures) from the one related to even cycles (2^{nd} query). Those data are resolution time, number of messages exchanged and the presence or absence of a solution for the query. The 1^{st} query (the one related to odd cycles) has been submitted when no overlay network has still emerged. Therefore, the only information available to each peer, for efficiently performing query propagation, is the one related to the connectivity graph. If the solution has been found, a corresponding overlay network is built at the end of the odd cycles, one superpeer emerges and the 2^{nd} query (which refers to the same goal of the first one) can be solved by taking advantage of the overlay network knowledge.

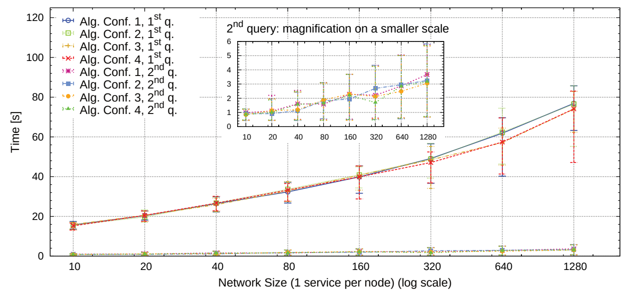
After each pair of cycles, the P2P network is brought back to the original configuration: the overlay network is cleared, superpeers are removed, and services are shuffled among peers, introducing a variance element at each pair of cycles to collect statistically more relevant data.

Together with the main query, to be solved by a 10-services composition and issued by one of the network peers, other nodes concurrently query the system for an unsolvable goal, in order to produce noise. In the figures, average values are reported together with min-max confidence intervals.

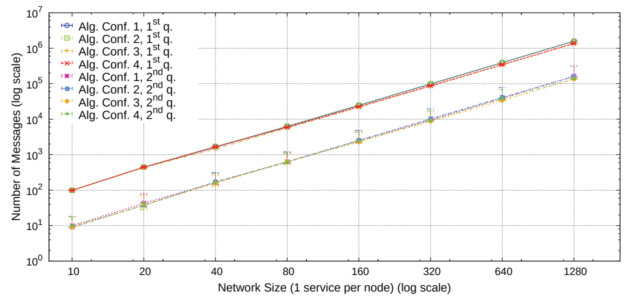
Fig. 4 is related to topology **(1)** and compares the case in which the goal is solved without the overlay network (1^{st} query) to the one in which the overlay network, built from the previous composition, is exploited (2^{nd} query).

The graphs in the figures demonstrate that, in topology **(1)** ($\approx size/10$ neighbors per node), with any of the algorithm configurations considered (Alg. Conf. from 1 to 4), the service composition overlay may be effectively reused for solving more quickly (lower curve in Fig. 4a) an already solved goal (higher curve in Fig. 4a). Resolution time for the 2^{nd} query is lowered by the presence, at the superpeer's repository, of the composite service solution previously found for the 1^{st} query. The presence of the composition overlay network also reduces the number of messages exchanged during the resolution of the 2^{nd} query (Fig. 4b), preserving high levels of recall. Fig. 4c only shows recall for the 1^{st} query. The 2^{nd} one, submitted in presence of the service composition overlay built after resolution of the 1^{st} query, has always been solved in our experiments (recall is 1, relatively to the number of solutions found for the 1^{st} query). In Fig. 4b, recall is always equal to the highest possible value **(1)** (with the exception of size 20, where the network is quite sparse, ≈ 2 neighbors per node).

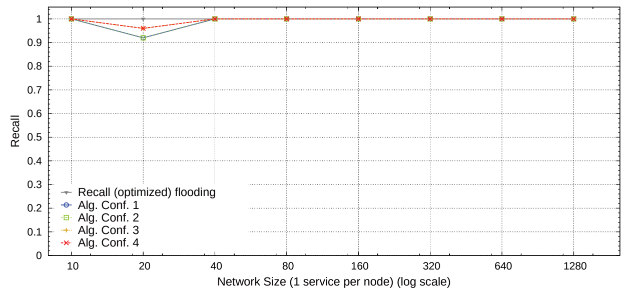
Figures 4d, 4e and 4f compare composition based on probabilistic forwarding with composition based on flooding. The graphs show the evident benefits of our probabilistic forwarding approach with respect to flooding and optimized flooding in relation to resolution time and number of mes-



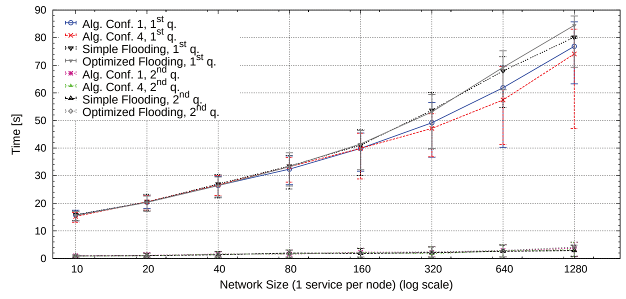
(a) Average resolution time - 1^{st} and 2^{nd} query



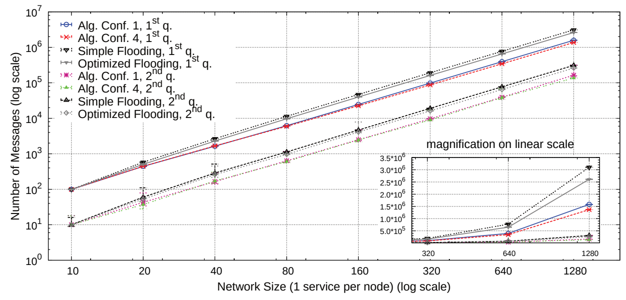
(b) Average exchanged messages - 1^{st} and 2^{nd} query



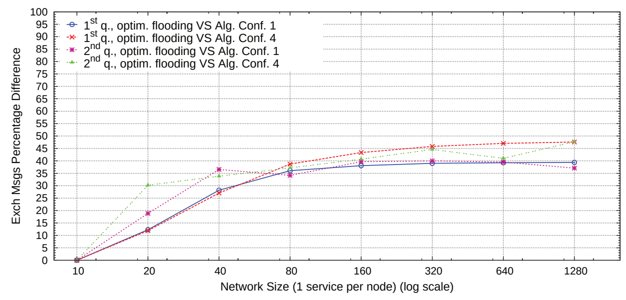
(c) Recall 1^{st} query



(d) Average resolution time - comparison with flooding and optimized flooding



(e) Average exchanged messages - comparison with flooding and optimized flooding



(f) Percentage message difference between opt. flooding and probalistic forwarding

Figure 4. Topology (1), 10-service P2P composition: performance evaluation

sages exchanged (both for the 1^{st} and for the 2^{nd} query).

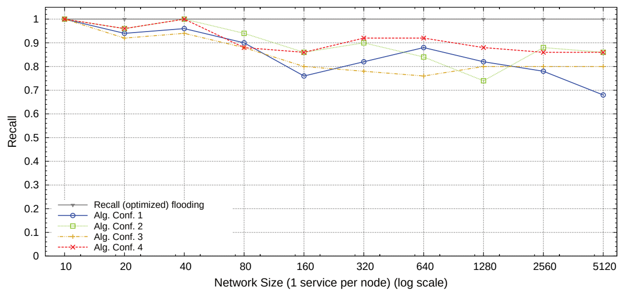
The same network configuration has been evaluated also in the case of topology **(2)**, obtaining good performance results as in the case of topology **(1)**. The left part of Fig. 5 shows a higher and more stable recall (close to 0.9 on large networks) when using an algorithm configuration with a higher coefficient for the hyperbole density evaluation function (specifically, Alg. Conf. 4 gives the best result). The right part of the figure shows percentage message reduction with respect to optimized flooding, when using probabilistic forwarding configurations, in case of both absence and presence of the service composition overlay network.

VII. Conclusion

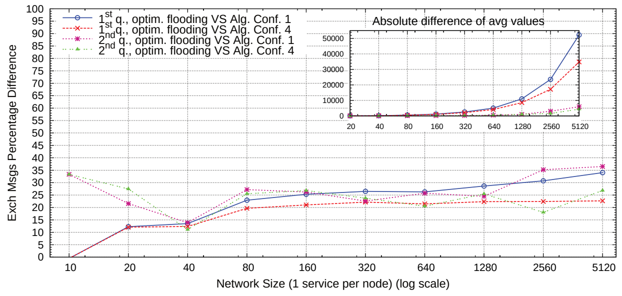
We proposed a technique to improve discovery and composition in P2P unstructured service networks, based on a probabilistic forwarding algorithm driven by the network knowledge, such as network density and semantic service grouping. The technique reduces the messages exchanged during discovery and composition relying on two considerations: if the network is dense, forwarding can be limited to a small number of neighbors; if the network is semi-structured in superpeers and peers, forwarding can be directed to the superpeers that may own the desired information.

The approach has been validated by using a simulator to observe resolution time, recall and message overhead on small and large size P2P networks. The experimental results show that, when using a traditional backward search approach, both the time for service composition and the number of messages exchanged are significantly reduced, while keeping almost unchanged the recall, especially when the network is dense.

We are currently working on improving the discovery and composition process, by using also a distributed bidirectional search. The expected benefit is twofold: first, it is possible to have concurrent searches in the P2P service network in both goal directions, reducing the response time when solutions are present; second, when there are no complete solutions for



(a) Recall 1^{st} query



(b) Percentage message difference between opt. flooding and probalistic forwarding

Figure 5. Topology (2), 10-service P2P composition: performance evaluation

a goal, gaps in partial found solutions can be identified and suggested to service providers as business opportunities.

References

[1] E. Zimeo, A. Troisi, H. Papadakis, P. Fragopoulou, A. Forestiero, and C. Mastroianni, "Cooperative self-composition and discovery of grid services in P2P networks," *Parallel Processing Letters*, vol. 18, no. 03, pp. 329–346, 2008.

[2] A. Forestiero, C. Mastroianni, H. Papadakis, P. Fragopoulou, A. Troisi, and E. Zimeo, "A scalable architecture for discovery and planning in P2P service networks," in *Grid Computing*, S. Gorlatch, P. Fragopoulou, and T. Priol, Eds. Springer US, 2008, pp. 97–108.

[3] L. Bevilacqua, A. Furno, V. di Carlo, and E. Zimeo, "A tool for automatic generation of ws-bpel compositions from owl-s described services," in *Software, Knowledge Information, Industrial Management and Applications (SKIMA), 2011 5th International Conference on*, sept. 2011, pp. 1–8.

[4] O. D. Sahin, C. E. Gerede, D. Agrawal, A. E. Abbadi, O. H. Ibarra, and J. Su, "SPiDeR: P2P-based Web Service Discovery," in *Proceedings of the Third International Conference Service-Oriented Computing (ICSOC 2005)*. Springer, Dec. 2005, pp. 157–169.

[5] Z. Zhengdong, H. Yahong, L. Ronggui, W. Weiguo, and L. Zengzhi, "A P2P-based Semantic Web Services Composition Architecture," in *e-Business Engineering, 2009. IEEE International Conference on*, oct. 2009, pp. 403 –408.

[6] D. Redavid, S. Ferilli, and F. Esposito, "P2P support for OWL-S discovery," in *Proceedings of the Mining Complex Patterns Workshop (MCP'11)*, 2011.

[7] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '01. ACM, 2001, pp. 149–160.

[8] P. Fragopoulou, C. Mastroianni, R. Montero, A. Andrjezak, and D. Kondo, "Self-* and adaptive mechanisms for large scale distributed systems," in *Grids, P2P and Services Computing*, F. Desprez, V. Getov, T. Priol, and R. Yahyapour, Eds. Springer US, 2010, pp. 147–156.

[9] M. Bisignano, G. D. Modica, and O. Tomarchio, "JaxSON: A Semantic P2P Overlay Network for Web Service Discovery." in *SERVICES I*, 2009, pp. 438–445.

[10] M. Gharzouli and M. Boufaida, "PM4SWS: A P2P Model for Semantic Web Services Discovery and Composition," *Journal of Advances in Information Technology*, vol. 2, no. 1, 2011.

[11] Gnutella Protocol Specification v. 0.4. The Gnutella Developer Forum. [Online]. Available: http://rfc-gnutella.sourceforge.net/developer/stable/index.html

[12] J. Risson and T. Moors, "Survey of Research towards Robust Peer-to-Peer Networks: Search Methods," *Computer Networks*, vol. 50, pp. 3485–3521, December 2006.

[13] G. Baryannis and D. Plexousakis, "Automated Web Service Composition: State of the Art and Research Challenges," ICS-FORTH, Tech. Rep., 2010.

[14] JXTA. Last accessed on November 2012. [Online]. Available: http://jxta.kenai.com/

[15] G. Tretola and E. Zimeo, "Structure Matching for Enhancing UDDI Queries Results," in *Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications*. IEEE Computer Society, 2007, pp. 21–28.

[16] E. Giallonardo and E. Zimeo, "More Semantics in QoS Matching," in *Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications*. IEEE Computer Society, 2007, pp. 163–171.

[17] Annual International Contest S3 on Semantic Service Selection. Last accessed on November 2012. [Online]. Available: http://www-ags.dfki.uni-sb.de/~klusch/s3/index.html

[18] M. Klusch and P. Kapahnke, "iSeM: Approximated Reasoning for Adaptive Hybrid Selection of Semantic Services," in *Proceedings of the 2010 IEEE Fourth International Conference on Semantic Computing*, ser. ICSC '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 184–191.

[19] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based Aggregation in Large Dynamic Networks," *ACM Trans. Comput. Syst.*, vol. 23, pp. 219–252, August 2005.

[20] A. Montresor and M. Jelasity, "PeerSim: A scalable P2P simulator," in *Proc. of the 9th Int. Conference on Peer-to-Peer (P2P'09)*, September 2009, pp. 99–100.