A Two-Tiered Approach to Enabling Enhanced Service Discovery in Embedded Peer-to-Peer Systems*

Antonio Brogi, Sara Corfini, and Thaizel Fuentes

Department of Computer Science, University of Pisa, Italy
{brogi,corfini,fuentes}@di.unipi.it

Abstract. Recent technology advances are pushing towards a full integration of low-capacity networked devices in pervasive embedded P2P systems. One of the challenges of such integration is to allow low-capacity devices both to invoke and to provide services, while featuring enhanced service discovery mechanisms that are necessary to automate service invocation in pervasive environments. In this paper we present a two-tiered approach to enabling enhanced service discovery in embedded P2P systems. We first present a super-peer based overlay network featuring a matching capability aware routing of messages, and saving the resource consumption of low-capacity devices while keeping the overall network traffic low. We then present a service discovery protocol that exploits such underlying overlay network to suitably distribute service contracts on devices capable of analysing them, thus enabling enhanced service discovery even in nets mainly formed by low-capacity devices. Finally, we discuss some experimental results that confirm the viability of the proposed approach.

1 Introduction

Recent advances in hardware and wireless technologies have paved the way for a full integration of low-capacity networked devices in pervasive embedded P2P systems. In this perspective, Service-oriented Computing [1] has proven to provide suitable abstractions to master the complexity of large applications. The notion of *service* is used to represent sets of functionalities offered by a peer[1], service providers publish into service registries *contracts* describing the provided services, while service consumers query service registries to locate the services they need to interact with.

To achieve truly automated pervasive systems, service discovery and invocation should be entirely automated, which means that enhanced service discovery mechanisms should be featured to reduce the possibility of failures in automated discover-and-invoke steps. In particular, one of the desired enhancements in the

* Research partially supported by EU FP6-IST STREP 0333563 SMEPP.

[1] In this paper we will use the terms "*peer*" and "*device*" interchangeably.

service discovery process concerns the quality of the results of the discovery process. Indeed while the need of including *signature information* in service contracts to enable interoperability is universally accepted (e.g., WSDL has prominently emerged as the de facto standard for defining the syntax of the functionalities featured by Web services), signature information is not enough to fully automate the discover-and-invoke step. For this reason, *ontological annotations* —to overcome non-relevant differences in the syntactic description of services— as well as *behavioural information* —to verify that service interactions will not lock— are starting to be included in service contracts of embedded P2P systems [2].

Achieving an effective implementation of enhanced service discovery is however one of the critical issues in pervasive embedded P2P environments for various reasons:

– Service registries cannot be centralised for obvious scalability and reliability reasons, thus service contracts have to be suitably distributed among the peers participating in the application.
– A distributed implementation of the service discovery process should aim at saving the resource consumption of low-capacity devices, which could otherwise consume all their resources by participating in the discovery protocol and then become unavailable when other peers will invoke the services they offered.
– Intuitively, contracts of services published by low-capacity devices should hence better be stored by higher-capacity devices. The implementation of such a policy is however complicated by the fact that devices storing contracts may (unexpectedly) disconnect from the network (e.g., because of mobility or battery exhaustion reasons). Also, not all devices are in general capable of analysing all types of information contained in service contracts.

Various service discovery architectures for (pervasive) P2P environments have been proposed over the last years. Those architectures are however typically tailored to efficiently deal with contracts and queries describing a specific type of information (e.g., [3,4,5] focus on syntactic information, while [6,7,8] focus on ontology-annotated queries), and they cannot be straightforwardly exploited to efficiently implement discoveries based on other types of information. Consider a query specifying both ontology-based and behaviour requirements. One could exploit for instance the approach of [6] to locate the service contracts matching the ontology-based requirements, and then check whether the partially matched services also satisfy the behaviour requirements of the query. The service contracts found by [6] may be however hosted by peers which do not feature behaviour-aware matching algorithms, and in those cases it would be necessary to discover other peers capable of performing a behaviour-aware analysis of contracts and to move the candidate contracts there to complete the matching. However, moving sets of contracts across a P2P network would seriously increase net traffic and severely affect the efficiency of the resulting discovery process.

In this paper we present a two-tiered approach to enabling enhanced service discovery (taking into account different types of information contained in service contracts) in embedded P2P systems for pervasive environments. We first present a super-peer based overlay network featuring a matching-capability aware routing of messages, capable of *(o1) saving the resource consumption of low-capacity devices* and *(o2) keeping the overall network traffic low.* We then present a service discovery protocol that exploits such underlying overlay network to *(o3) suitably distribute service contracts on devices capable of analysing them*, thus enabling enhanced service discovery even in nets mainly formed by low-capacity devices. Finally, we discuss some experimental results to assess the level of achievement of objectives *(o1)*, *(o2)*, and *(o3)* and to confirm the viability of the proposed approach.

The proposed overlay network is a slight extension of the classical super-peer model [9], where a set of connected (*super*) peers acts as servers for the rest of (*client*) peers. One of the novelties of our overlay network is the introduction of the notion of *assistant* peers, which can provide functionalities (matching functionalities, in our context) not provided by the super peers. While assistant peers may not own enough resources to play the role of super peers, these can exploit assistant peers to provide the functionalities they cannot provide by themselves. A suitable *ranking function* is introduced to rank peers (and classify them as client, assistant or super peers) with respect to the provided matching functionalities and their physical resources.

The rest of the paper is organised as follows. Sections 2 and 3 present the overlay network and the service discovery protocol, respectively. Section 4 discusses some optimisations which mainly concern the maintenance of the overlay network. Some experimental results are discussed in Section 5, while related work is discussed in Section 6. Finally, Section 7 presents some concluding remarks.

2 Overlay Network

As anticipated in Section 1, the overlay network proposed in this paper slightly extends the classical super peer model by introducing the notion of *assistant* peers (Fig. 1). Intuitively, an assistant peer is a peer which provides some (matching) functionalities, yet not owning enough physical resources to be a super peer. A super peer can exploit assistant peers in its vicinity to provide its client peers with functionalities it cannot provide by itself. In order to classify a peer as super, assistant or client peer we introduce a ranking function ρ which ranks peers by taking into account the features (i.e., physical resources and provided functionalities) described in peer advertisements.

Specifically, an advertisement A_P of a peer P is a tuple

$$A_P = \langle WL_P, CPU_P, RAM_P, MOB_P, POW_P, MF_P \rangle$$

where $WL_P \in [0..1]$ denotes the current workload of the peer in terms of both the number of contracts stored by P and the requests managed in the last time interval (i.e., 1 denotes an overloaded peer, 0 an idle peer), CPU_P and RAM_P respectively denote the CPU speed and the RAM capacity of the peer, $MOB_P \in$ {stationary, moving} describes whether the peer is moving or not, $POW_P \in$ {power plugged, on battery} describes whether the peer is plugged to the power

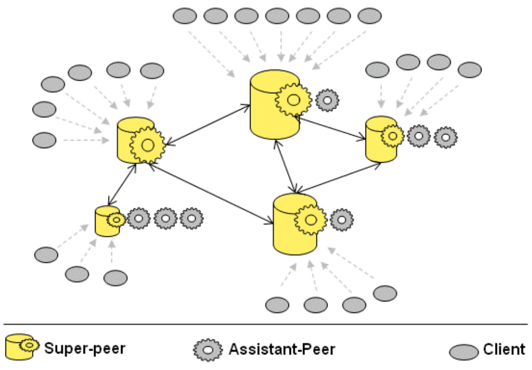


Fig. 1. Overlay network topology

or not, and $MF_P = MF_{self(P)} \cup MF_{assistants(P)}$ denotes the functionalities that the peer provides either by itself ($MF_{self(P)}$) or through its assistant peers ($MF_{assistants(P)}$).

The functionalities MF_P described in a peer advertisement are precisely the *matching functionalities* that the peer is able to provide. A matching functionality is a *type* of matching algorithm. The group of matching functionalities that we consider are ⟨*syntactic*⟩, ⟨*syntactic, ontological*⟩, ⟨*syntactic, light-behavioural*[2]⟩, ⟨*syntactic, behavioural*⟩, ⟨*syntactic, ontological, light-behavioural*⟩ and ⟨*syntactic, ontological, behavioural*⟩.

A ranking function $\rho : \mathcal{ADV} \times \mathbb{N} \longrightarrow \mathbb{R}$ is used to rank peers. A peer Q ranks a peer P by computing the value $\rho(A_P, D_{PQ})$, where A_P is the advertisement of P and D_{PQ} is the distance (i.e., number of physical hops) from P to Q. Intuitively, when ranking peers featuring similar physical resources, ρ ranks higher peers providing more matching functionalities. On the other hand, peer that advertise no matching functionalities (viz., $MF_P = \emptyset$) are always ranked 0, independently of their physical resources.The values computed by ρ are also inversely proportional w.r.t. the second parameter (i.e., $\rho(A_P, D_{PQ}) > \rho(A_P, D_{PQ} + \varepsilon)$) to take into account the physical vicinity among peers.

A peer Q can exploit ρ to classify a peer P (possibly itself) as client, assistant or super peer, as follows:

- P is a *client* peer if $\rho(A_P, D_{PQ}) = 0$,
- P may act as *assistant* peer if $\rho(A_P, D_{PQ}) > 0$, and
- P may act as *super* peer if $\rho(A_P, D_{PQ}) > t$

where t is a threshold to establish whether a peer can be a super peer or not, and $D_{PQ} = 0$ if $Q = P$. As illustrated in Fig. 1, client peers connect to a (single)

[2] We distinguish *light-behavioural* matching algorithms (checking the "may"-compatibility of service protocols, i.e., the existence of at least one successful interaction trace) from full *behavioural* matching algorithms (checking the full compatibility of service protocols, i.e., that all interaction traces are successful).

nearby super peer, while – differently from the classical super peer model – a super peer may exploit assistant peers in its vicinity to provide the matching functionalities that it cannot provide by itself.

2.1 Network Maintenance

A peer Q acting as super peer maintains a list of *fingers* to other super peers in its vicinity and a list of its current *assistants*. Both lists contain tuples of the form $\langle A_P, D_{PQ}, t\rangle$, where t is the time at which Q received advertisement A_P from P. A peer that does not act as super peer maintains only one super peer tuple, corresponding to the chosen super peer.

The core functionality of the overlay network can be summarised as follows:

- Each peer enters the net as a *client* peer, and it remains idle until it receives a routing request (from one of its upper layer protocols) or it receives a message from some other peer.
- When a client peer C needs to route a message, if it has not yet chosen its super peer then it broadcasts to its vicinity a SuperPeerDiscovery message carrying its advertisement A_C.
- When a peer P receives a SuperPeerDiscovery message from C:
  - If P is acting as super peer then P unicasts its advertisement A_P to C.
  - If P is not acting as super peer but $\rho(A_P, 0) > t$ or $\rho(A_P, D_{CP}) \geq \rho(A_C, D_{CP})$ then P unicasts A_P to C and self promotes itself to super peer.
- When a client peer C receives an advertisement A_P from P at time t:
  - If C does not have a super peer and $\rho(A_C, 0) < \rho(A_P, D_{PC})$ then C chooses P as its super peer and stores $\langle A_P, D_{PC}, t\rangle$ as its *super* tuple.
  - The same happens if C has a super peer $S \neq P$ but $\rho(A_P, D_{PC}) > \rho(A_S, D_{SC}) + \Delta$
  - If P was already the super peer chosen by C then C simply updates its *super* tuple into $\langle A_P, D_{PC}, t\rangle$.

  In any case, if C provides some matching functionality which is not provided by P, then C unicasts its advertisement A_C to P.
- A client C which has broadcasted a SuperPeerDiscovery message and which has not received any advertisement A_P such that $\rho(A_C, 0) < \rho(A_P, D_{PC})$ self promotes itself to super peer after a timed wait.

Super peers periodically broadcast their advertisements to their vicinity and unicast their *fingers* list to the super peers in such a list. When a super peer receives an advertisement or the list of another super peer, it updates its own *finger* and *assistant* lists by exploiting the ranking function ρ. Note that a super peer can be chosen as an assistant by another super peer.

2.2 Message Routing Protocol

The objective of the overlay network is to deliver messages to the peers which provide the required matching functionalities. The overlay network routes each message with respect to an associated key $k = \langle MF^S, MF^G \rangle$, which specifies the set MF^S of matching functionalities which *must* be strictly provided by the target peer, and the set MF^G of the matching functionalities which should be greedily provided by the target peer.

If the sender of the message is a client peer, the message is first routed to the super peer of the sender (if any, otherwise the sender peer broadcasts a SuperPeerDiscovery message to choose a super peer). If the target super peer provides the required matching functionalities, the message and – possibly – the link(s) to the assistant peer(s) necessary to satisfy $\langle MF^S, MF^G \rangle$, are dispatched to the upper service discovery layer. The super peer may also forward the message to those super peers in its fingers table which match the received key k. The radius of such forwarding is set by the service discovery layer.

3 Service Discovery Protocol

The service discovery layer features a service discovery protocol by storing service contracts in super peers and by exploiting matching functionalities provided by super and assistant peers to match contracts with queries.

The service discovery protocol exploits the overlay network to publish and to search for service contracts by passing a publication or query message and a key $k = \langle MF^S, MF^G \rangle$ to the overlay network. As described in the previous section, the key specifies the matching functionalities that the target peer(s) must provide (viz., MF^S) and should greedily provide (viz., MF^G). For instance, a message associated with the key $\langle \{ \langle$*syntactic, light-behavioural*$\rangle \}, \{ \langle$*syntactic, ontological, light-behavioural*$\rangle \} \rangle$ will be received by super peers capable of performing (possibly with the help of their assistants) both *syntactic* and *light-behavioural* matching and optionally, also ontological matching.

When a (service discovery) message and the associated key k are received by a target super peer, the overlay network dispatches the message, and the link(s) to the assistant peer(s) possibly necessary to satisfy the matching functionalities $\langle MF^S, MF^G \rangle$, to the service discovery layer. If the message is a publication message the (discovery layer of the) target super peer stores the contract of the published service. If the message is a query, the (discovery layer of the) target super peer first matches the contracts that it stores locally. If the super peer can provide all the required matching functionalities MF^S by itself, it returns the matched contracts to the peer that generated the request, otherwise, it forwards the query and the (partially matched) contracts to (some of) its assistant peers. Assistant peers match the received contracts by executing the matching functionalities requested by k, and return the matched contracts the peer that generated the request.

As anticipated in the previous section, the service discovery layer can specify the radius of the forwarding of messages among super peers. A peer can hence decide for instance to publish and search services only in its super peer.

Summing-up, the service discovery protocol (SDP) publishes and searches service contracts by invoking the overlay network (ON). To do this, the former invokes the later with calls of the form: route(m,k,forwardHops) where m is either publish(contract) or query(contractTemplate) and where k=$\langle MF^S, MF^G \rangle$. Suppose that the service discovery protocol of a peer Q invokes such a call. Then the behaviour of the overlay network of Q can be synthesised as follow:

```
If $Q$ is not super peer Then {
  If $\nexists$ super peer Then $Q$ starts discovering a super peer
  <m,k,forwardHops> is sent to the ON of the super peer of $Q$
}
Else { // this branch is also the code to be executed when a ON component receives a
       // message from another ON component
  If $MF_Q \supseteq MF^S$ Then {
    If $MF_{self(Q)} \not\supseteq MF^S$ AND m=query(ct) Then
      m is dispatched to the SDP (of $Q$), together with a subset $H$ of the assistants of
      $Q$ such that $(\cup_{h \in H} MF_h \cup MF_{self(Q)}) \supseteq MF^S \wedge$
                    $\forall h \in H\ (\cup_{k \in H \setminus \{h\}} MF_k \cup MF_{self(Q)}) \not\supseteq MF^S$
    Else m is dispatched to the SDP (of $Q$)
    dispatched = true
  }
  Else dispatched = false
  // to forward message m
  If (dispatched is false) OR (forwardHops > 0) Then {
    If (dispatched is true) Then forwardHops=(forwardHops−1)
    <m,k,forwardHops> is sent to all $R$ in the fingers of $Q$ such that $MF_R \supseteq MF^S$
  }
}
```

4 Optimisations

To simplify the reading, in Section 2 we have presented the core aspects of our overlay network. There are, however several important optimisations that have been implemented to reduce the number of messages exchanged for network maintenance and routing.

– *Limited broadcast.* We have seen that super peers periodically broadcast their advertisement. To control the number of potential clients, super peers dynamically update the radius of such broadcast according to their own current workload and available resources.
– *Passive mode.* If an (active) super peer does not receive routing message for a while, it switches to *passive mode* and stop periodically broadcasting its advertisement – until it will receive a routing message and switch back to active mode.
– *Checking the aliveness of super peers.* Whenever a peer routes a message to a super peer A, it firstly checks the time t_A when it received the advertisement from A. If t_A is up-to-date (i.e., the advertisement has been received recently), the message is sent *asynchronously* to the super peer. Otherwise, if t_A is out-of-date, the message is sent *synchronously* to the super peer, in

order to get an acknowledgement from it. If an acknowledgement is received, t_A is updated, otherwise A is not considered a valid super peer any more.
– *Avoiding network partitioning.* Network partitioning may especially occur in networks of mobile, limited-resource devices, where super peers advertise themselves in a short vicinity. To avoid that, whenever a peer chooses a new super peer, it notifies its old super peer (if any) of the availability of the new super peer, thus facilitating the inter-connection among super peers.

5 Evaluation

In order to assess the viability of the proposed overlay network and service discovery protocol, we analysed their behaviour with PlanetSim[10], an extensible discrete-event Java simulator for key-based routing protocols in P2P overlay networks.

We run a set of simulations for networks populated by heterogeneous peers, randomly distributed and moving in a $600 \times 600m^2$ area and capable of communicating within a $100m$ range. The matching functionalities provided by each peer were obtained according to the peer's randomly generated capabilities, and peers could unexpectedly leave the network due to battery exhaustion. In all the simulations, first all peers join the network, then 40% of peers (randomly chosen) started publishing service contracts, and then 60% of peers (randomly chosen) started issueing queries to discover services. The simulations were run by scaling the number n of peers from 10 to 150 (with a pace of 10), and for each n the result was obtained by taking the average of the results of 15 different tests run for 200 units of simulation time. In each test 20% of peers (randomly chosen) were high-capacity devices and 10% of peers (randomly chosen) were mobile devices, (randomly) moving in their vicinity during the entire simulation.

The first set of simulations was run to assess the degree with which the proposed service discovery protocol accomplished objective *(o3)* set in the Introduction, namely "*to suitable distribute service contracts on devices capable of analysing them*".

The metric we used to measure the accomplishment level of *(o3)* was the percentage of published contracts matching a query q that were successfully located by the service discovery protocol on devices capable of analysing them.

Formally, let q be a query, let k be the key used to route q and let $MF^S(q)$ be the set of required matching functionalities specified with q. Then for each query q we computed the ratio:

$$\frac{\sharp\{c_h \mid \exists P : P \ stores \ c_h \wedge h \bowtie k \wedge q \ hits \ P \wedge MF_P \supseteq MF^S(q)\}}{\sharp\{c_h \mid \exists P : P \ stores \ c_h \wedge h \bowtie k\}} \tag{1}$$

where c_h denotes a contract that was published with key h, $h \bowtie k$ denotes that the key h and k match, and MF_P denotes the set of matching functionalities provided by peer P (possibly with the help of its assistants).

Fig. 2 illustrates the results of the simulation, with only 1-hop routing forwarding among super peers. We can observe that even with a little percentage

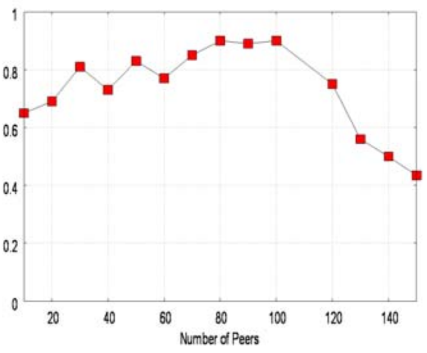


Fig. 2. Testing the ability of locating contracts on devices capable of analysing them

of high-capacity devices (20%), the accuracy of the discovery is very high up to 100 peers. After that it starts to decrease because of the incompleteness inherent to the super peer model (exacerbated here by considering only 1-hop routing forwarding among super peers).

The second set of simulations was run to assess the degree with which the proposed overlay network accomplishes objective *(o1)* set in the Introduction, namely "*saving the resource consumption of low-capacity devices*".

The first metric we used to measure this was the percentage of (overlay) messages received by low-capacity devices w.r.t the overall number of messages exchanged due to routing activities (Fig. 3(a)). We compared our proposal with a basic implementation of the super-peer model (similar to [3][3]) and with the implementation of Chord DHT[11] provided by PlanetSim, customised to support mobility and to fit our statistics outputs.

We observed in Fig. 3(a) that, while Chord does not take into account device capabilities, when the number of peer grows, our proposal reduces the percentage of messages received by low-capacity devices w.r.t the basic super peer implementation.

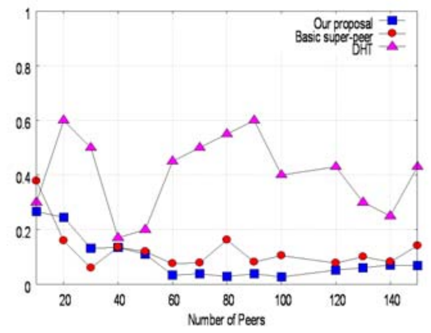
The saving of resource consumption of low-capacity devices achieved by our proposal is even better highlighted in Fig. 3(b), where the used metric is directly the number of low-capacity devices still alive[4] at the end of the simulation.

A further set of simulations was done to assess the degree with which the proposed overlay network accomplishes objective *(o2)* set in the Introduction, namely "*keeping the overall network traffic low*".

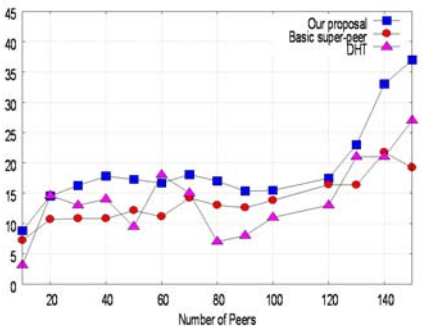
We first measured the number of (overlay network) messages generated by network maintenance activities. We can observe in Fig. 4(a) how the optimisations implemented in our proposal (Section 4) allow to reduce the network

[3] Super peer advertises right after joining the net, maintains a registry of their clients, and client requests are routed to super peers with compatible (numeric) keys.

[4] The simulation decrements the battery of a device every time it receives a message at the physical level (either for network maintenance or for routing).

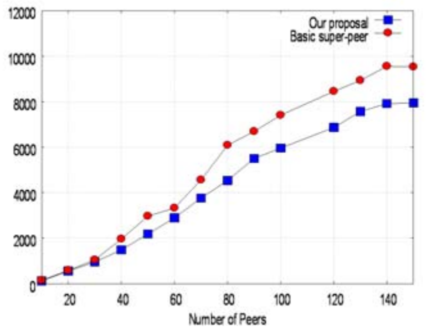


(a) Percent of messages received by low-capacity devices during routing

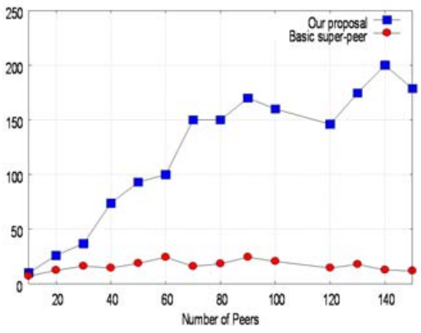


(b) Number of low-capacity devices alive at the end of the simulation

Fig. 3. Testing resource consumption of low-capacity devices



(a) Traffic due to network maintenance



(b) Traffic due to routing

Fig. 4. Testing network traffic

traffic generated for network maintenance by the basic super peer model. The numerical values for Chord –which generates only $O({\log_2}^2(N))$ messages during network maintenances[11]– are not plotted in Fig. 4.

We then measured the number of (overlay network) messages generated by routing activity. Fig.4(b) shows that our proposal generates, as expected, quite more traffic for routing than the basic super peer implementation. The reason for this is that the implementation of our overlay network used in the experimentation routes messages with the objective of hitting devices providing the desired matching functionalities, but it does not take into account the other information included in contracts and queries (whose analysis is to be entirely performed by the upper service discovery level). A more fair comparison of the two approaches should consider an implementation of our overlay network capable of exploiting such information to reduce the number of fingers and assistants to which messages are routed. Such an implementation could be obtained by allowing the

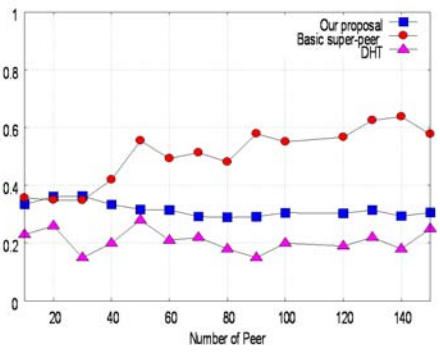


Fig. 5. Overlay/physical messages ratio

service discovery level to suitably configure the type of filters to be exploited by the overlay network. This is precisely one of our planned future works.

In order to get an estimation of the actual traffic generated at the physical level, we measured the ratio (Fig. 5) between the number of messages (both for routing and network maintenance) at the overlay level and the corresponding messages at the physical level[5]. Fig. 5 shows that the ratio of our proposal is better that Chord (which is not topology-aware), but worse than the basic super-peer model, as our ranking function ρ privileges the availability of (matching) functionalities to physical vicinity.

6 Related Work

To overcome the serious limitations —scalability and reliability (single point of failure)— of the first P2P architectures that relied on a centralised server (e.g., like in Napster's original design), a number of decentralised architectures have been proposed for P2P systems. These can be roughly partitioned into unstructured, structured and semi-structured architectures.

A main drawback of unstructured architectures (like Gnutella[12], JXTA[13,14]) is message explosion, caused by the use of message flooding to route messages. Moreover, each peer –target of a message routing– executes its own matching algorithm(s) without exploiting enhanced matching algorithms possibly provided by higher-capacity peers. This makes unstructured architectures unsuitable to implement enhanced service discovery mechanisms for embedded P2P systems.

Structured architectures, such as Distributed Hash Tables (DHTs [4,5]), sensibly reduce the number of (overlay) messages. However, the unawareness of the underlying physical topology and of peers'capacities make DHTs unsuitable to implement enhanced service discovery mechanisms for embedded P2P systems.

[5] The simulator determined the number of physical messages corresponding to an overlay message sent from A to B by counting one physical hop every $100m$ over the Euclidean distance between A and B.

Also, DHTs are not particularly well-suited for mobile environments, where frequent unexpected (dis)connections would cause frequent costly reorganisation of the overlay network. It is worth mentioning [15], an extension of Pastry DHT [16] which takes into account static physical capabilities of peers (viz., cpu speed, ram, etc.), but does not consider dynamic properties like vicinity, workload or battery consumption, and [17], which extends Chord's identifiers [11] to take into account any type of information regarding the service provider. Both [15] and [17] however present the other general drawbacks of DHTs.

Semi-structured architectures (like our proposal) set up a backbone of super peers which act as mini-servers for the other peers in the network. While they do not present the drawbacks of unstructured and structured approaches, network partitioning and cyclic message forwarding may occur in semi-structured approaches like [18], where peers autonomously choose their links to other peers. [7] and [19] build an overlay network among peers "sharing common interests" (e.g., semantic concepts or types of services). In [7] peers are organised in clusters, each mapping a semantic concept and storing "similar" files, while queries are forwarded to clusters that feature compatible concepts. In [19], each cluster has a coordinator, coordinators are linked one another, and coordinators do not take into account the physical capabilities of peers. Data-centered approaches like [7] and [19] are however tailored to handle specific types of data, and they cannot straightforwardly exploited to efficiently implement discoveries based on other types of information, as we already mentioned in the Introduction. In [20] super peers exchange a hierarchical XML representation of the data they store and use path expressions to process incoming queries. Such an approach cannot be however exploited in networks consisting of low-capacity devices only, because of the resources needed to process path expressions.

In [21] and [22], super peers constitute a DHT. In [21] low-capacity devices connect to a bootstrap node, which is chosen as super peer. If the chosen super peer is overloaded, the client is redirected to a non-overloaded super peer in the (Chord) ring, and service publication and discovery is based on keywords. In [22] peers discover and connect to super peers via JXTA protocols [13], and peers discover services by sending trivial semantic-based queries (a single taxonomy of types of services is considered) to super peers. Both [21] and [22] are not however well-suited for mobile networks of low-capacity devices, and they suffer from the previously discussed general drawbacks of structured architectures.

The approaches [23,24,25,3,6] are the most related with our proposal. In designing our architecture, we followed the guideline of [9] on how super peers can be selected, and we extended the concrete, yet partially defined, super peer network protocol of FastTrack [23] (inspired by `Kazaa, http://www.kazaa.com`). [24] ranks peers considering their static and dynamic capabilities. Super peers are dynamically elected in order to keep bounded the ratio between the number of clients peers and the number of super peers. Each super peer periodically compares its ranking against the ranking of its clients (and vice-versa). If the number of highest-ranked client peers exceeds a predefined threshold, then the super peer downgrades to client peer while the best ranked client peer promotes as super peer. In [25] super-peers are elected considering mainly the distance (measured as communication latency). Moreover, [25] deactivates super peers when they do not register clients, but (active) super peers advertise themselves even if they are not receiving queries from their clients. Instead, our proposal deactivates super peers when they do not receive service discovery messages for a while, thus saving network traffic. Differently from [24] and [25], we choose super peers by taking into account their matching functionalities mainly, but also our super peers do not register their clients to save memory consumption. Differently from [25], our proposal also dynamically sets the advertisement radius of super peers (i.e., the radius within which the super peers advertise themselves) with respect to their current capabilities, helping to keep bounded the ratio between client peers and super peers, similar to [24]. In the approaches [3] and [6], peers are organised into a semi-structured network similar to [25]. Super peers store service contracts and match them syntactically [3] and semantically [6]. The main novelty of our proposal with respect to [3] and [6] is that our architecture supports any matching approach, strengthened by the introduction of assistant peers, which are the key ingredient to implement an efficient and accurate service discovery mechanism capable of matching any type of query.

Last, but not least, we mention [26] that provides a high-level service discovery architecture enabling the coexistence of different contracts languages and (legacy) matching algorithms and allowing low-level communication among different (multi-radius) networks.

Different contract and query description languages and different matching algorithms are used in pervasive environments, ranging from syntactic [5], to semantics-based [8,27], to behaviour-based [28] service discovery. [26] provides a high-level service discovery architecture enabling the coexistence of such (legacy) matching algorithms. Our proposal can be integrated as a (vertical) middle-layer in multi-tiered architectures like[26]: We locate –and route messages to– devices capable to perform required matching functionalities, abstracting on the underlying multi-radius network –provided by [26]– and perimetrically w.r.t. both the contract (query) languages and the (legacy) matching algorithms, which will be locally provided by systems like [26].

7 Concluding Remarks

We have presented a two-tiered approach to enabling enhanced service discovery in embedded P2P systems. As we have seen, the proposed service discovery protocol exploits an overlay network featuring a matching capability aware routing of messages to suitably distribute service contracts on devices capable of analysing them, thus enabling enhanced service discovery even in nets mainly formed by low-capacity devices. We have also analysed the collected experimental data to assess the level of achievement of the three objectives that we set in the Introduction —*(o1) saving the resource consumption of low-capacity devices*, *(o2) keeping the overall network traffic low*, and *(o3) suitably distributing service contracts on devices capable of analysing them*— yielding a confirmation of the viability of the proposed approach.

Our plans for future work include to integrate first in our overlay network key-based data filters (such those employed in [6,8]) to drive message routing, as we already mentioned in Section 5. Then we plan to develop a full-fledged service discovery system where existing (both ontology-based and behaviour-aware) matchers can be plugged-in, so as to be able to start a thorough assessment of the versatility of the proposed overlay network and a comparative assessment at the service discovery level.

References

1. Papazoglou, M.P., Georgakopoulos, D.: Service-Oriented Computing. Communications of the ACM 46(10), 24–28 (2003)
2. Benigni, F., Brogi, A., Corfini, S., Fuentes, T.: Contracts in a Secure Middleware for Embedded Peer-to-Peer Systems. In: Proc. of the 2^{nd} Workshop on Formal Languages and Analysis of Contract-Oriented Software (FLACOS) (2008)
3. Sailhan, F., Issarny, V.: Scalable Service Discovery for MANET. In: 3^{rd} IEEE Int. Conf. on Pervasive Computing and Communications (PerCom), pp. 235–244. IEEE Computer Society, Los Alamitos (2005)
4. Lua, E.K., Crowcroft, J., Pias, M., Sharma, R., Lim, S.: A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. IEEE Communications Surveys and Tutorials 7(2), 72–93 (2005)
5. Louati, W., Zeghlache, D.: SPSD: A Scalable P2P-based Service Discovery Architecture. In: IEEE Wireless Communications and Networking Conference (WCNC), pp. 2588–2593 (2007)
6. Mokhtar, S.B., Preuveneers, D., Georgantas, N., Issarny, V., Berbers, Y.: EASY: Efficient semAntic Service discoverY in pervasive computing environments with QoS and context support. Journal of Systems and Software 81(5), 785–808 (2008)
7. Garcia-Molina, H., Crespo, A.: Semantic Overlay Networks for P2P Systems. Stanford InfoLab, Technical Report 2003-75 (2003)
8. Skoutas, D., Sacharidis, D., Kantere, V., Sellis, T.K.: Efficient Semantic Web Service Discovery in Centralized and P2P Enviroments. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 583–598. Springer, Heidelberg (2008)
9. Yang, B., Garcia-Molina, H.: Designing a Super-Peer Network. In: Proc. of the 19^{th} Int. Conf. on Data Engineering (ICDE), pp. 49–60. IEEE Computer Society, Los Alamitos (2003)
10. Pujol Ahulló, J., García López, P., Sànchez Artigas, M., Arrufat Arias, M., París Aixalà, G., Bruchmann, M.: PlanetSim: An extensible framework for overlay network and services simulations. Universitat Rovira i Virgili, Tech. Rep. DEIM-RR-08-002 (2008)
11. Stoica, I., Morris, R., Karger, D.R., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM), pp. 149–160 (2001)
12. Gnutella team, Gnutella discovery protocol, `http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf`
13. JXTA team, Jxta specification, `https://jxta-spec.dev.java.net/`
14. Srirama, S.N., Jarke, M., Zhu, H., Prinz, W.: Scalable Mobile Web Service Discovery in Peer-to-Peer Networks. In: 3^{rd} Int. Conf. on Internet and Web Application and Services (ICIW), pp. 668–674. IEEE Computer Society, Los Alamitos (2008)

15. Liang, Q.A., Chung, J.-Y., Lei, H.: Service Discovery in P2P Service-oriented Environments. In: Proc. of the 8^{th} Int. Conf. on E-Comemerce Technology and of the 3^{rd} Int. Conf. on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE). IEEE Computer Society, Los Alamitos (2006)
16. Rowstron, A., Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In: Guerraoui, R. (ed.) Middleware 2001. LNCS, vol. 2218, pp. 329–350. Springer, Heidelberg (2001)
17. He, Q., Yan, J., Yang, Y., Kowalczyk, R., Jin, H.: Chord4S: A P2P-based Decentralised Service Discovery Approach. In: IEEE Int. Conf. on Services Computing, pp. 221–228. IEEE Computer Society, Los Alamitos (2008)
18. Kobayashi, H., Takizawa, H., Inaba, T., Takizawa, Y.: A Self-Organizing Overlay Network to Exploit the Locality of Interests for Effective Resource Discovery in P2P Systems. In: Proc. of the 2005 Symposium on Applications and the Internet (SAINT), pp. 246–255. IEEE Computer Society, Los Alamitos (2005)
19. Doulkeridis, C., Nørvåg, K., Vazirgiannis, M.: DESENT: decentralized and distributed semantic overlay generation in P2P networks. IEEE Journal on Selected Areas in Communications 25(1), 25–34 (2007)
20. Thilliez, M., Delot, T.: A Localization Service for Mobile Users in Peer-to-Peer Environments. In: Crestani, F., Dunlop, M.D., Mizzaro, S. (eds.) Mobile HCI International Workshop 2003. LNCS, vol. 2954, pp. 271–282. Springer, Heidelberg (2004)
21. Hofstätter, Q., Zöls, S., Michel, M., Despotovic, Z., Kellerer, W.: Chordella – A Hierarchical Peer-to-Peer Overlay Implementation for Heteregeneous, Mobile Environments. In: 8^{th} Int. Conf. on Peer-to-Peer Computing (P2P), pp. 75–76. IEEE Computer Society, Los Alamitos (2008)
22. Ayorak, E., Bener, A.B.: Super Peer Web Service Discovery Architecture. In: Proc. of the 23^{rd} Int. Conf. on Data Engineering (ICDE), pp. 1360–1364. IEEE, Los Alamitos (2007)
23. FastTrack team, FastTrack protocol, `http://cvs.berlios.de/cgi-bin/viewcvs.cgi/gift-fasttrack/giFT-FastTrack/PROTOCOL?revision=1.19`
24. Xiao, L., Zhuang, Z., Liu, Y.: Dynamic Layer Management in Superpeer Architectures. IEEE Trans. on Parallel and Distributed Systems 16(11), 1078–1091 (2005)
25. Jesi, G.P., Montresor, A., Babaoglu, O.: Proximity-Aware Superpeer Overlay Topology. IEEE Tran. on Network and Service Management 4(2), 74–83 (2007)
26. Caporuscio, M., Raverdy, P.-G., Moungla, H., Issarny, V.: ubiSOAP: A Service Oriented Middleware for Seamless Networking. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 195–209. Springer, Heidelberg (2008)
27. Zhou, G., Yu, J., Chen, R., Zhang, H.: Scalable Web Service Discovery on P2P Overlay Network. In: IEEE Int. Conf. on Services Computing (SCC), pp. 122–129. IEEE Computer Society, Los Alamitos (2007)
28. Shen, Z., Su, J.: Web Service Discovery Based on Behavior Signatures. In: Proc. of the 2005 IEEE Int. Conf. on Services Computing (SCC), pp. 279–286. IEEE Computer Society, Los Alamitos (2005)