The International Journal of Parallel, Emergent and Distributed Systems
Vol. 00, No. 00, January 2009, 1–21

RESEARCH ARTICLE

Service Provisioning For a Next-Generation Adaptive Grid

Amos Brocco, Béat Hirsbrunner
Department of Informatics, University of Fribourg, Switzerland
(*Received 00 Month 200x; in final form 00 Month 200x*)

Finding ways of providing efficient, scalable and robust grid service provisioning is one of the foundations of next generation grid systems. Recognizing the need for flexible, autonomic, and self-manageable solutions, we propose a flexible grid service provisioning framework built on a two layer architecture that promotes a clear separation of concerns. A low-level layer manages a self-organized peer-to-peer overlay that provides grid membership and fundamental message exchange mechanisms. The key differentiating factor of the proposed approach is the use of collaborative bio-inspired algorithm to maintain and optimize the overlay topology in order to improve communication efficiency, as well as adapting to dynamic network conditions. Building on these premises, a high-level layer provides grid services such as resource discovery and monitoring. The proposed solution is evaluated under different scenarios that assess its qualities concerning performance, adaptiveness, reliability, fault-resilience and network communication efficiency. Analysis of the obtained results serves as a guideline for the selection of optimal resource discovery and multicast strategies.

1. Introduction

As high performance computing systems as well as high bandwidth connections become more common throughout the world, the possibility of exploiting a large pool of geographically dispersed computational resources has become a reality for solving complex tasks. Finding ways of managing such systems, thereby making it possible to efficiently access their resources, is one of the key issues that existing research on *grid computing* focuses on [1]. In particular, the possibility of discovering available resources should be considered as an essential factor for a full exploitation of a grid. The provisioning of efficient, scalable and robust resource discovery services becomes therefore a necessity for a successful deployment of next-generation grid systems. Furthermore, such architectures need to be flexible, autonomic, and self-manageable, in order to reduce deployment costs, increase reliability, and meet dynamic users' needs [2]. These additional goals strongly motivate further research on emerging and enabling technologies for grids.

Resource discovery in typical grid systems, such as [3–5], is provided by centralized indexing services, known as grid resource information services [6] or monitoring and discovery systems [6]. On one side, centralized solutions enable deterministic information retrieval with limited bandwidth consumption [7]. On the other side, this approach limits the scalability of grid systems and represents a communication bottleneck as well as a single point of failure. A partial solution to this problem is delivered by hierarchical system designs: the grid is managed by multiple indexing services, themselves organized as a hierarchy that typically reflects the inherent organization of grid hosts. At each level of the hierarchy, index services aggregate the information gathered from lower levels [8]. Search queries are propagated upward

*Email: amos.brocco@unifr.ch, beat.hirsbrunner@unifr.ch

ISSN: 1744-5760 print/ISSN 1744-5779 online
© 2009 Taylor & Francis
DOI: 10.1080/1744576YYxxxxxxxx
http://www.informaworld.com

in the architecture, starting from the lowest level, up to the point that a match has been found. Although hierarchical information systems improve the scalability of the grid, the inherent reliability problems of centralized designs remain, albeit at a smaller scale.

The evolution of distributed systems toward P2P networks gave rise to a multitude of novel solutions that could also be applied to grid systems in order to bring a significant increase in both flexibility and fault-resilience [9]. There exist different examples of peer-to-peer based information systems for grids [10–12]. Depending on the underlying topology and data organization, it is possible to classify these systems as structured and unstructured.

Structured systems [13, 14] are typically based on deterministic topologies, and organize the information in a distributed hash table according to a key-node match. Accordingly, the topology itself encompasses important semantic information enabling very efficient and deterministic resource discovery, with low network overhead and response time. Unfortunately, one of the major drawbacks of hash indexing is the fact that it does not inherently support range search and multidimensional queries. Although these issues have been partially addressed [15, 16], characteristics of queries and resources in a grid system may nonetheless exacerbate the expected efficiency of search in structured approaches [17]. Additionally, structured overlays may not be well suited for dynamic conditions and suffer from excessive churn rates [18]. Despite these limitations, several projects focus on implementing grid resource discovery upon structured overlays [19, 20].

On the contrary, unstructured systems [21, 22] do not enforce strict topologies, and are more tolerant toward nodes joining and leaving the network. Resource discovery is traditionally performed using flooding mechanisms, by initially posting the query to a local set of nodes and then propagating queries progressively throughout the network. Flooding allows for free text queries but may suffer from excessive network communication overhead, unless some structure is introduced in the overlay (e.g. *super-peers* [23]), data is replicated across peers [24] or clustered [25], or optimized flooding strategies are employed [26–28].

The cornerstone of our research has been to enable efficient grid resource discovery on top of an optimized peer-to-peer overlay retaining some of the aforementioned advantages of both structured and unstructured systems, while overcoming some of their limitations. In order to achieve the successful integration of grid and P2P systems, it is nonetheless important to recognize their diverse functional requirements. Accordingly, we propose a flexible and self-managed framework built on a two layer architecture consisting of a self-organized peer-to-peer overlay and a grid service provider. Membership management as well as basic group communication primitives are provided at the overlay level, and provide a solid foundation to deliver high-level grid services such as resource discovery, load balancing, and monitoring. Self-organization enables efficient low-level communication by dynamically adapting logical connections between nodes in order to reduce average path length in the overlay. This optimization of the network is accomplished by means of a completely decentralized approach based on a collaborative bio-inspired algorithm, and takes full advantage of its inherent adaptive and robust behavior. The conceptual design of the proposed layered architecture leads to a clear separation of concerns that hides the complexity of the underlying network and furthermore facilitates high level access to grid resources.

The rest of this paper is organized as follows: Section 2 presents a brief overview of search techniques in peer-to-peer networks and examples of grid service provisioning architectures based on unstructured peer-to-peer overlays. Section 3 introduces the middleware architecture, whereas Section 4 details the operation of the network

communication layer and the overlay management algorithm. Section 5 discusses the service provisioning layer and the services available to applications. Sections 6 and 7 present the evaluation scenarios, respectively the results obtained. Finally, Section 8 summarizes the work presented in the paper and provides some insights to future developments.

2. Related Work

As pointed out in [29], the underlying differences between a grid and a P2P system make it necessary to consider different performance analysis strategies when solutions devised for the latter are applied on grids. Traditional methods for resource discovery in P2P systems necessitate adjustments to better fit the design goals of a grid. In particular, improved query forwarding strategies that are more tailored for the characteristics and dynamics of grids need to be considered. Grid systems are typically more stable than P2P networks, as the set of available resources shared in a grid changes more dynamically, depending on actual scheduling constraints. Resource discovery in grids is also different than in P2P: searching for a file requires an exact match, whereas searching for a computing resource mainly involves range queries.

This section provides a review of state-of-the-art algorithms and techniques that enable efficient grid service provisioning and communication over P2P overlays. Because our work deals with a self-structured overlay, a detailed discussion of structured systems is omitted: the interested reader is referred to [30] for a survey of additional P2P resource discovery mechanisms.

The underlying concept behind decentralized resource discovery assumes that the node requesting some resources sends its query to some other known node (typically its neighbors in the topology). Recipients determine if they can fulfill the request and eventually respond to the initiating node. If the request cannot be fulfilled, the query is further forwarded to one or more nodes [31].

According to a taxonomy proposed in [30], search methods for unstructured networks can be classified as either *uninformed* or *informed*. Uninformed search methods just rely on stochastic or deterministic query forwarding logic: no semantic information about the query being processed is used during the forwarding process.

The primary uninformed technique for searching in unstructured networks is *flooding* [26]. Flooding involves sending the query to all neighbors, and having each recipient node propagate the query further in the overlay, up to a given distance specified by a time-to-live (TTL) counter. The performance of simple flooding protocols is heavily influenced by the topology of the overlay, and it may be difficult to control the generated traffic (mostly caused by the high number of duplicated messages) [24]. To limit the traffic explosion, methods like *iterative deepening* (also known as *expanding ring*) or *probabilistic flooding* [32] can be employed. In iterative deepening queries are first executed with a small TTL value, and repeated with increasing TTL if no results are found. With probabilistic flooding, queries are forwarded only to a small random subset of all neighbors.

Another technique that has been proposed is *random walks* [24, 33]. Instead of forwarding a query to all neighbors, at each step only one random path is chosen. A random walk query is commonly called *walker*: the maximum of steps a walker can complete determines the depth of the search. Random walks consume less bandwidth than flooding, but are subject to larger delays [34]. For this reason, to improve response time several random walks can be started in parallel. Furthermore, combined search techniques employing both flooding and random walks can be implemented [34]. The efficiency of the these search protocols can be fur-

ther improved by replicating data across the overlay [35], by avoiding unnecessary retransmission of data packets, or by clustering information [36].

An alternative approach is to use an informed search method, which relies on certain heuristics to drive query forwarding. These heuristics are usually based on previous experience [31, 37, 38], and allow for improved bandwidth usage because queries are forwarded only to nodes that are more likely to return a positive answer.

Research on resource discovery in unstructured overlays does not only concentrate on improved search schemes. Another field of interest focuses on characterizing overlay networks and identifying how topologies influence the efficiency of resource discovery [39, 40]. Recognizing that some topologies are better suited for decentralized resource discovery, several projects focused on adding structure (e.g. *super-peers* [23]) or reorganizing overlays in an autonomous and adaptive way [41, 42]. We are particularly interested in the second approach, as our work reflects the idea of creating a self-structured overlay to improve the efficiency of existing search methods as well as of other services such as group communication.

In this context, ACQUAINTANCES [37], proposes a mechanism for augmenting a GNUTELLA-like overlay with additional links based on previous experience. Connections between nodes are created or removed according to the number of positive responses. Although very similar to informed search approaches, ACQUAINTANCES shows that it is possible to further improve the efficiency of resource discovery by modifying the overlay topology. Similarly, PROSA [43] builds and maintains a P2P network based on social relationships and acquired semantic information about peers.

A further case of exploiting a dynamically adapting topology is P-GRID [44]. P-GRID is also an example of a self-organized system, utilizing a structure similar to a distributed hash table to index information. The system implements a load balancing mechanism that ensures fair replication of data across the nodes. On top of the overlay, self-organizing services such as peer identity management and trusted communication have been implemented.

A more complex solution is represented by UMM [45], which uses a two layer architecture separating the tasks of maintaining an overlay and disseminating the information in an efficient way. Connections between nodes are optimized both to reduce latency and to increase available bandwidth. Furthermore, UMM utilizes an optimized query forwarding scheme that tries to avoid message duplication and consequently restrain traffic overhead.

In the same spirit, SAXONS [46], maintains an overlay with low latency, high bandwidth paths, as well as small distances. The overlay provides efficient multicast communication across the overlay that can be exploited to deliver higher level services.

Finally [47] proposes an example of service-oriented framework, SP2A, based upon a peer-to-peer overlay and focusing on resource sharing in grids.

Our work follows the path traced by the aforementioned approaches, in that it focuses on constructing and maintaining an optimized overlay that enables the implementation of efficient higher level services such as resource discovery or multicast communication. In this respect, the most important contribution of our work is the use of a collaborative bio-inspired algorithm to manage the overlay.

3. Framework Architecture

The proposed framework architecture, as initially introduced in [48], is composed of two functional layers: a low-level network communication layer, and a higher level service provisioning layer (Figure 1).

This layered architecture establishes a clear separation of concerns regarding communication between peers and high level grid service provisioning. Accordingly, the network communication layer is responsible for membership and communication management, by maintaining proper network connectivity and implementing basic unicast and multicast mechanisms. Conversely, the grid service provisioning layer exposes higher-level functionalities such as resource discovery and querying. This layer is comprised of a Grid Services Provider and an Aggregator component: the former provides support for simple tasks like resource discovery queries, while the latter enables complex multi-task queries as well as caching of retrieved results for improved response time.

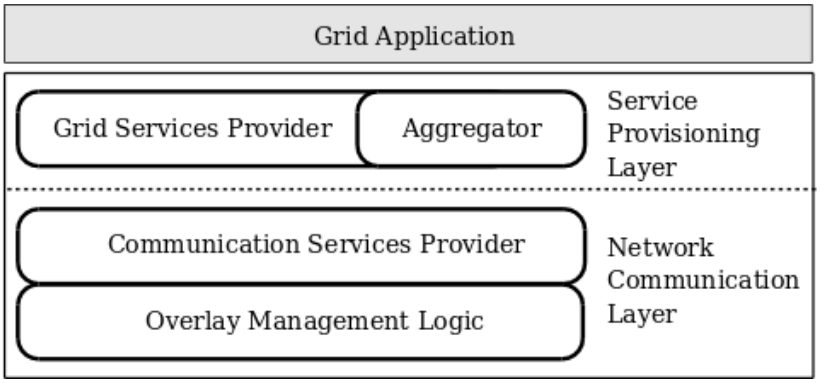

Figure 1. Framework Architecture

Applications interface with the upper layer while remaining totally unaware of the complexity of the network management. Typical interaction between an application and the grid service provisioning layer involves submitting resource discovery queries and retrieving the corresponding results.

The proposed framework is not concerned with scheduling of jobs on remote nodes nor with actual communication with remote resources: as these tasks typically depend on complex scheduling and resource access policies, they are left to the grid application. Consequently, in a grid deployment scenario communication between applications and the framework will be mediated by a scheduling and resource management middleware, as proposed in [49].

4. Network Communication Layer

Two tasks are amongst the responsibilities of the network communication layer: low level connectivity between grid nodes (also known as *membership management*), and support for the exchanging of messages across the overlay. In line with the aforementioned separation of concerns, this layer is conceptually composed of two functional components: an overlay management logic, and a communication services provider. We discuss here the details of both components.

4.1 Overlay Management Logic

The overlay management logic ensures connectivity by maintaining an optimized overlay with minimal average path length between nodes. The management algorithm, called BlåtAnt-S, is an improved version of our previous algorithm BlåtAnt-R [50] that reduces both the complexity and the generated traffic of the latter, while retaining its performance characteristics. A brief discussion of the

differences between these two versions of the algorithm is provided at the end of this section. BLÅTANT-S is fully distributed, adaptive in respect to changes in the underlying network, and fault tolerant. In particular, its functionalities are implemented using different types of lightweight mobile agents traveling across the network. The behavior of these mobile agents is inspired by swarm intelligence [51].

Swarm intelligence algorithms mimic the behavior of insect colonies in order to solve optimization problems, such as routing problems in dynamic networks. In this respect, the proposed overlay management algorithm autonomously organizes the topology exploiting the synergy of different agents behaving like an ant colony. For this reason, we refer to agents implemented in our proposed algorithm as *ants*.

Each node n_i on the grid maintains a set of logical neighbors N_i, and a fixed-size cache table with information about other nodes, such as their estimated distance and their neighbors. This table is updated using the information carried by incoming ant agents, and provides a local partial view of the network. Furthermore, the number of neighbors cannot exceed a user-defined limit, to prevent hub formation.

Topology Optimization Rules. According to its own partial information about the network, each node performs local optimization of its connections by creating new logical links and removing existing ones. This process is controlled by two rules exploiting a user defined parameter D, such that the resulting average path length is between D and $2D-1$.

Connection Rule. Let n_i and n_j be two non-connected nodes in an overlay G, and $d_G(n_i, n_j)$ the minimal routing distance from n_i to n_j in G. A new link between n_i and n_j is created if the following condition holds:

$$d'_G(n_i, n_j) \geq 2D - 1 \tag{1}$$

where $d'_G(x, y)$ is defined as $min(d_G(x, y), d_G(y, x))$, and the logical connection is created by adding n_i to N_j, respectively n_j to N_i.

The Connection Rule reduces the average path length between each pair of nodes to a value less than $2D-1$. Conversely, as the number of connections per node is bounded, the Disconnection Rule is applied in order to remove links that are not deemed necessary to achieve the desired average path length.

Disconnection Rule. Let n_i and n_j be two connected nodes in an overlay network G, $i \neq j$. Let $G' \leftarrow G \setminus \{n_i\}$ and N_i the set of neighbors of n_i. Node n_i is disconnected from $n_j \in N_i$ if:

$$\exists\, n_k \in N_i, k \neq j, |N_j| > |N_k| \;:\; d^*_{G'}(n_j, n_k) + 1 \leq D \tag{2}$$

where $d^*_G(x, y)$ is defined as $max(d_G(x, y), d_G(y, x))$, and the disconnection consists of removing n_i from N_j, respectively n_j from N_i.

Proof of Correctness. The presented rules ensure that the average path length, as well as the diameter (upper bound for all average path lengths), in the resulting network satisfies the property previously enounced. We now formally prove that, with a global knowledge and in a finite number of steps, the application of these rules on an overlay network G leads to an average path length less than $2D-1$.

LEMMA 4.1 **Convergence of the Connection Rule** *Let be* $G_0 = G$ *and* G_{n+1} *the*

topology obtained by applying the Connection Rule to G_n for two randomly chosen nodes. Then $\exists\ k \geq 0,\ \forall\ i \geq 1\ :\ G_k = G_{k+i}$, *i.e.* $\forall\ n_i, n_j \in G_k\ :\ d_{G_k}(n_i, n_j) < 2D - 1$.

Proof Since the function on the distances resulting from the application of the Connection Rule is monotonously decreasing, for k big enough, then the result follows. ■

We now prove that creation of a new link by the *Connection Rule* cannot trigger the *Disconnection Rule*, as well as that the *Disconnection Rule* cannot partition the overlay.

LEMMA 4.2 **Monotonicity of Connection Rule** *Let G' be an overlay obtained by applying the Connection Rule to G. Let $\delta(G)_{n_i,n_j}$ be the condition that the Disconnection Rule is applicable in the overlay G for nodes n_i, n_j. Then*

$$\nexists\ n_i, n_j \in G\ :\ \delta(G)_{n_i,n_j} \Longrightarrow \nexists\ n_i, n_j \in G'\ :\ \delta(G')_{n_i,n_j}$$

Proof Let (n_i, n_j), $n_i, n_j \in G$, the link created by the Connection Rule. Because $d_G(n_i, n_j)$ has minimal length, the smallest cycle in G' has a length $l \geq 2D-1+1 = 2D > D$. ■

LEMMA 4.3 **Connected Overlay Preservation** *The Disconnection Rule cannot partition a connected overlay G.*

Proof Let $n_i, n_j, n_k \in G$, such that $n_j, n_k \in N_i$. If there exists a path between n_j and n_k satisfying the condition set by the *Disconnection Rule*, there are at least two disjoint paths from n_i to both n_j and n_k in G. Thus, removing a single connection cannot partition G. ■

COROLLARY 4.4 **Convergence of Disconnection Rule** *Given an overlay G, let be $G_0 = G$, and G_{n+1} the overlay obtained by applying the Disconnection Rule to G_n for two randomly chosen nodes. Then* $\exists\ k \geq 0,\ \forall\ l \geq 1 : G_k = G_{k+l}$.

THEOREM 4.5 *Given an overlay G, let be $G_{0,0} = G$, and $G_{n,m}$ the overlay obtained by applying, in any order, n times the Connection Rule and m times the Disconnection Rule on $G_{0,0}$ for randomly chosen nodes. Then* $\exists\ k \geq 0, \exists\ l \geq 0$, *and* $\nexists\ n_i, n_j \in G_{k,l}$ *such that the Connection Rule or the Disconnection Rule apply.*

Proof Follows directly from Lemmas 4.1, 4.2, 4.3, and Corollary 4.4. ■

The presented proof only considers the situation where global and precise information about the overlay is available. In a fully distributed scenario each node must rely on partial and potentially out-of-date information about the overlay. Whereas degraded information about path distances in the overlay just increase the average path length and result in a less optimized overlay, concurrent application of the *Disconnection Rule* may lead to a partitioning. To ensure that the algorithm works in a fully distributed scenario a restriction on the *Disconnection Rule* is introduced: for each considered cycle, only the node with the greatest identifier (according to some ordering known to all nodes) is allowed to perform a disconnection. That node is referred as to the *master* of a given cycle. We now prove that under this assumption, even a completely decentralized application of the disconnection rule cannot partition the overlay.

LEMMA 4.6 **Safeness of the Disconnection Rule** *Concurrent application of the disconnection rule cannot lead to a partitioning of the network.*

Proof We first consider the situation where complete knowledge of the overlay is

available. Because in each cycle only the corresponding master can remove a link, the concurrency problem is completely avoided and the overlay cannot be partitioned. In a fully distributed scenario local information on each node may be outdated, e.g. refer to cycles that have already been broken. It is thus necessary to prevent multiple disconnection on the same cycle: this can be achieved by maintaining a history of detected and broken cycles. This solution has the potential drawback of requiring a large amount of storage on each master node, especially in very dynamic networks. Another solution is to only allow a master to remove connections with its own neighbors, thus making it possible to verify whether the cycle has already been broken by relying only on already available local and up-to-date information. ■

Ant Species. Different tasks of the algorithm are performed using various species of ant agents. Ants execute on nodes, and each species differs from others by the information it carries, and the behavior it triggers on each visited node:

- *Discovery Ants* wander across the network building up a fixed-size vector v of visited node identifiers. This information is processed by each visited node to update the local view and evaluate both the connection and the disconnection rules.
- *Construction-Link Ants* carry connection requests from a node wanting to join the overlay. If the recipient cannot accept the request (because the maximum number of neighbors has been reached), the ant is forwarded to the neighbor with the least degree. *Construction-Link Ants* are also used to recover connectivity when nodes quit the overlay.
- *Optimization-Link Ants* perform connections between two nodes as result of the Connection Rule.
- *Unlink Ants* disconnect two nodes according to the Disconnection Rule.
- *Update Neighbors Ants* notify a node when one of its neighbors has updated its neighbors set. This ensures that each node is able to recover from abrupt disconnection of a neighbor by connecting with its last known neighbors.
- *Ping Ants* are used to keep connections between nodes alive.

Discovery Ants are responsible for continuously monitoring the state of the network, and are thus periodically generated on each node according to a uniform probability distribution. After a fixed number of hops traveled in the overlay, *Discovery Ants* are disposed. In contrast, other species of ants are instantiated as necessary to fulfill non-repetitive tasks (for example, connecting two nodes). As the optimization task depends on the information gathered by *Discovery Ants*, running the algorithm with an empty population will prevent any improvement of the topology.

Recovery Procedure. When a node disconnects from the network it has to inform all of its neighbors and initiate a recovery procedure that will ensure connectivity is preserved. This procedure involves sending out *Construction-Link Ants* to all neighbors and connecting them using a ring topology (Figure 2). In the event of an abrupt disconnection, this procedure is started by every neighbor node that detects the failure.

Pheromone Trails. Communication between real ants occurs using a stigmergic (i.e. indirect) mechanism which involves leaving chemical *pheromone* trails in the environment. These chemical traces can be sensed by other individuals in the colony and their concentration indicates the desirability of a given path. With time, unless new chemical is left by an insect, the concentration of the

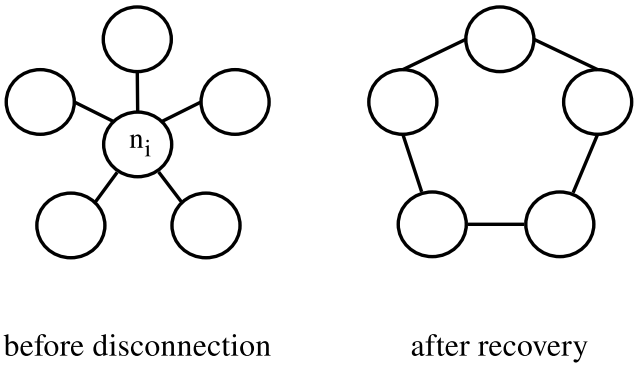

Figure 2. Recovery after node n_i leaves the network

trail completely evaporates. Evaporation has the added benefit of seamlessly suppressing errors and overcoming bad system decisions. In our system, we emulate this phenomenon, and in that respect pheromone concentrations are represented as numerical values $\tau \in [0, 1]$ stored on each node. Ants executing on a node can both read the actual concentration of a trail, and *reinforce* it by increasing its value up to a maximum of 1.

Each node periodically simulates *evaporation* by lowering the value of a trail τ according to an update function $\tau \leftarrow \tau * \psi$, and $\psi < 1$. We distinguish between incoming β trails, and outgoing γ trails. When an ant travels from node n_i to a neighbor n_j, the corresponding trail $\gamma_i[n_j]$ on n_i is reinforced. Conversely, when the ant arrives on n_j, pheromone trail $\beta_j[n_i]$ is reinforced. *Discovery Ants* wandering around the network preferably chose paths with lower γ pheromone concentration, ensuring a fair coverage of the network. Furthermore, abrupt node disconnection can be detected by monitoring the concentration of β trails: when values approach a lower threshold a recovery procedure is started. If application traffic is low, the trail between two nodes may not be frequently reinforced, and thus completely evaporate even though the corresponding nodes are still connected to the overlay. To prevent this from happening, *Ping Ants* are regurarly deployed as soon as trail concentration falls below a certain threshold.

Rules Evaluation. *Discovery Ants* relay their information vectors v to each visited node. A node n_i receiving a vector v updates the cache table, and then proceeds by determining if the *Disconnection Rule* applies. The information vector is traversed and the distance d_v (absolute indices difference) between each pair of identifiers n_j, n_z with $z \neq j \ \wedge \ n_j, n_z \in N_i$ is computed. If $d_v < D$, a disconnection procedure is started with the neighbor (either n_j or n_z) with greater degree (number of links). To evaluate the *Connection Rule*, the distances of each node n_z in the local cache table are sorted, and a connection procedure is started with the node at the greatest distance $> 2D - 1$. If a connection procedure has already been started, the following node is chosen.

Comparison to BlåtAnt-R. The former version of the presented algorithm, BlåtAnt-R, shares many similarities with the presented S version. Information collected by *Discovery Ants* is more complete in the R version, including both the identifier of the node being visited, and the identifiers of its neighbors. This enables a more complete partial view of the network and a more precise evaluation of distances by enabling the construction of a partial graph at the expense of a greater computational complexity and increased network traffic. In particular, instead of deriving distance estimations from the relative positions

of identifiers in v, a shortest-path algorithm [52] is used.

4.2 Communication Services Provider

The communication services provider implements basic data packet delivery services to grid applications by exploiting the overlay maintained by the network management layer. Three different message delivery models are implemented: unicast (point-to-point message transmission), multicast (one-to-many message transmission), and broadcast (one-to-all message transmission).

Both multicast and broadcast communications are performed using flooding algorithms. With flooding, each received message is forwarded to all neighbors, up to a pre-defined distance from the initiating node (number of hops in the overlay). Unrestricted flooding on unstructured network suffers from excessive bandwidth consumption and is thus inadequate; in particular, redundant paths result in a large number of unnecessary packet forwardings. To improve the efficiency of the system, and limit the amount of traffic generated by flooding, probabilistic and limited flooding strategies may be employed. Probabilistic flooding [32] (also known as *teeming* [26]) means that, at each step, the message is forwarded only to a random subset of all neighbors. With limited flooding each node avoids forwarding messages that have already been processed in the past, which implies that nodes keep track of received messages. Accordingly, broadcast communication is provided with a *best-effort* policy: although the depth of flooding can be specified, depending on the actual overlay topology, only a subset of the nodes may be contacted. The efficiency of this broadcasting strategy is validated by experimental results which are presented in Section 7.

4.3 Name Resolution

Communication across the overlay can either refer to precise IP addresses, hostnames that can be translated using an external DNS (Domain Name System) resolution service, or application defined logical names. At the lowest level, communication is always performed using IP addresses. If no prior knowledge is available, application defined logical name resolution is achieved in a fully distributed manner by broadcasting a request and waiting for a reply from the concerned host. Each node may implement a caching mechanism that helps improving the efficiency of such distributed resolution method.

5. Service Provisioning Layer

The Service Provisioning Layer provides applications with a high-level interface to the set of resources shared in the grid. The service provisioning layer is not tied to a single application, and its services may be concurrently accessed by multiple applications by means of asynchronous message passing and callback notifications. A similar communication mechanism is used to interact with the overlay management layer, both to deliver messages on the network, and to receive incoming messages from other peers. Moreover, the Service Provisioning Layer also exposes communication services that enable best-effort communication within nodes: this can be used, for example, to broadcast service messages to individual node administrators.

Amongst the tasks of the service provisioning layer is the processing of high-level application queries, and their translation to low-level grid messages handled by the overlay management layer. Application queries mostly involve requests for resources

in order to distribute computationally intensive jobs, as well as for monitoring their status. In that respect, resource discovery is supported by broadcast and multicast communication, whereas resource monitoring typically relies on point-to-point communication, unless multiple resources are concerned.

Additionally, this layer also provides a mechanism to match incoming queries from other nodes that are relayed as messages from the lower layer. An Aggregator component supports the grid services provider in dealing with complex resource discovery queries, and by providing cached results.

When a high-level query is received by the Grid Service Provider, it is first delegated to the Aggregator. The Aggregator component will first dissect complex queries, that may require the composition of different services, into different simple queries. For example, to support a load-balancing activity, not only does the application require a particular resource profile, but also it needs to choose the best match according to the available bandwidth. In this case, the Aggregator will decompose the task by first performing a resource discovery, and then profiling the available bandwidth on the connection to matching nodes. The Aggregator also helps improving response time by providing cached results from previous resource discovery queries. The cache is also used to profile searches and detect similarity between queries in order to issue proactive queries that will enhance results and reduce latency for popular requests.

If a query cannot be satisfied by cached results from the Aggregator, it is returned to the Grid Services Provider and a network wide resource discovery is commenced. Results from this operation will be processed by both the Grid Services Provider, in order to return them to the application, and by the Aggregator, to update its cache.

5.1 Resource Profiles

Nodes participating to the grid must publish a profile of available resources to the Grid Services Provider, in order to enable the matching of queries from the network. Although there exist different resource profile description languages (for example JSDL [53], GLUE [54], GRSL [55], etc.), actual implementation of our framework makes use of a custom dictionary of $\langle key, value \rangle$ pairs. Despite its simplicity, this representation mechanism enables both exact key matching as well as range searches.

6. Evaluation

Our research goal is to provide optimized resource discovery and group communication to support grid operations. In that respect, we evaluated the proposed framework in the context of a custom built simulation environment. Different scenarios were considered, to assess the performance of our system in terms of scalability, reliability and efficiency. This section details the experimental configuration of each scenario, as well as the assumptions that we have undertaken regarding the simulations. It should be noted that the resource discovery and group communication results presented in this paper have been obtained using an improved version of the framework presented in [48]. In particular, fine tuning optimizations were made in the forwarding mechanisms that result in more efficient flooding.

Simulation Environment. For practical reasons all experiments are run

in a discrete time simulator; nonetheless the implementation of the system remains fully distributed. We distinguish two types of timing: a global simulation timing, and local node timing. Global simulation timing controls the simulation of network dynamics, such as connection of new nodes to the overlay and disconnections. It is driven by migrations of the population of ants: at each iteration any ant may travel at most for one hop in the overlay (migrating from a node towards another node). On the contrary, local timing on each node depends on a logical timestamp that is updated according to incoming and outgoing ant traffic: the timestamp is incremented by one each time an ant arrives on the node, as well as when it leaves. This timing controls node's activity such as pheromone evaporation and creation of *Discovery Ants*. We assume that communication within the overlay is reliable; however the transport layer does not ensure a FIFO behavior, as the order of multiple messages originating from the same source node is not guaranteed at recipients.

Communication Costs. To evaluate the amount of traffic generated by the algorithm, thus determining the resulting network overhead, an estimation of the size of each data packet used by the framework has been made. In particular, for the overlay management part, the following estimations have been made:

- *Discovery Ant*: 420 bytes *plus* 24 bytes/visited node;
- *Construction-link Ant*: 444 bytes;
- *Optimization-link Ant*: 420 bytes;
- *Unlink Ant*: 420 bytes;
- *Update Neighbors Ant*: 420 bytes *plus* 24 bytes/neighbor.
- *Ping Ant*: 420 bytes;

Concerning the resource discovery and group communication tasks, the size of queries, replies and messages is considered as follows:

- *Resource discovery query*: 1024 bytes;
- *Resource discovery query reply*: 456 bytes;
- *Group communication message*: 1024 bytes;

All estimations include the actual payload as well as the size of a minimal UDP packet. A resource discovery query reply is sent to the initiating node for every match found.

Overlay Management. Unless otherwise specified, in all evaluation scenarios the main algorithm parameters are kept constant.

The choice of parameter values is based on well established results obtained in our previous research [50], and will be only briefly discussed here. In this respect, a sensitivity analysis of the algorithm parameters highlighted that minor variations do not significantly affect the qualitative behavior of the presented results. The pheromone decay ratio ψ is set to 0.991: according to our experience, this value provides an optimal evaporation rate for the considered network dynamics. Each node generates a new *Discovery Ant* with a probability of 1% every 150 iterations: this ant may subsequently travel for at most 50 hops in the overlay. Each *Discovery Ant* carries a vector of size 20 with the identifiers of the most recently visited nodes. The optimization parameter D is set to 5 in all considered evaluation scenarios: accordingly, the expected average path length in the resulting overlays is between 5 and 9.

Simulation Scenarios. Different aspects of the framework are evaluated, such as scalability, reliability and efficiency, and in that respect a variety of

network simulation scenarios were considered. All topologies are constructed starting from a pool of 10 nodes (referred to as *well-known* connection points), to which a number of additional nodes connect as soon as the simulation starts. To prevent the emergence of hubs, each node may accept a maximum of 8 connections. In all scenarios, as soon as the construction phase has been completed, the overlay size is in the range of 1281 nodes. The only exception is in the expanding network scenario, used to evaluate the scalability of the overlay, where the number of nodes is progressively increased from 10 to around 10000 by adding a node every 5 iterations. Within this scenario, the performance of the overlay management algorithm is also evaluated by varying the *Discovery Ants* birth probability from 0% (i.e. no optimization), to 1%, and 5%.

To assess the reliability of the overlay as well as the amount of network traffic generated under typical conditions, a set of dynamic scenarios have been considered. In these scenarios the overlay is periodically modified by dynamically simulating addition and removal of nodes following a stochastic Poisson process.

Concerning the reliability of the network, the considered connection and disconnection periods are of 20, 50, and 100 iterations. Disconnection of a node is achieved according to two equiprobable policies: either proper disconnection, where the departing node notifies its neighbors and reconnects them to preserve connectivity, or abrupt disconnection. Because the average rates of connections and disconnections are equal, the size of the network is kept almost constant.

By means of a comparative scenario, we also assess the qualities of BlåtAnt-S in respect to its predecessor, BlåtAnt-R. In particular, we measured the average path length for both algorithms in a dynamic network of 1281 nodes with a connection and disconnection period of 50 iterations.

Finally, to evaluate resource discovery and group communication a dynamic network scenario with a connection and disconnection period of 25 iterations is used.

Resource Discovery and Group Communication. The metrics used to determine the efficiency of resource discovery are the hit rate, as well as the forwarding strategy costs. In the context of grid systems, resources refer to the computational capabilities of the nodes themselves, thus the size of the considered resource pool is equal to the size of the network. The hit rate is defined as the percentage of successfully discovered resources out of all possible matching ones. In this respect, we assumed a replication ratio of about 2%, resulting in an average of 27 replicas for each possible resource shared on the network. Additionally, the forwarding strategy cost is measured as the total traffic generated by a resource discovery query based on the aforementioned communication costs.

Resource discovery queries are generated every 25 iterations. A set of 5 queries is concurrently started on randomly chosen nodes and broadcasted on the network to search for a resource shared by other nodes. Each query has a limited time-to-live, travels up to a maximum distance from the starting node in the overlay, and is transmitted according to a selective forwarding policy.

Different forwarding strategies are evaluated by varying the forwarding distance (TTL) as well as the maximum number of neighbors to which the query is forwarded at each step (FW). Once a match is found, a notification is sent to the initiating node, and the query is not further propagated to neighbors. In line with a limited flooding approach, forwarding is also stopped when a node receives a query that has already been processed before. In our evaluations no response time limit was set for resource discovery queries.

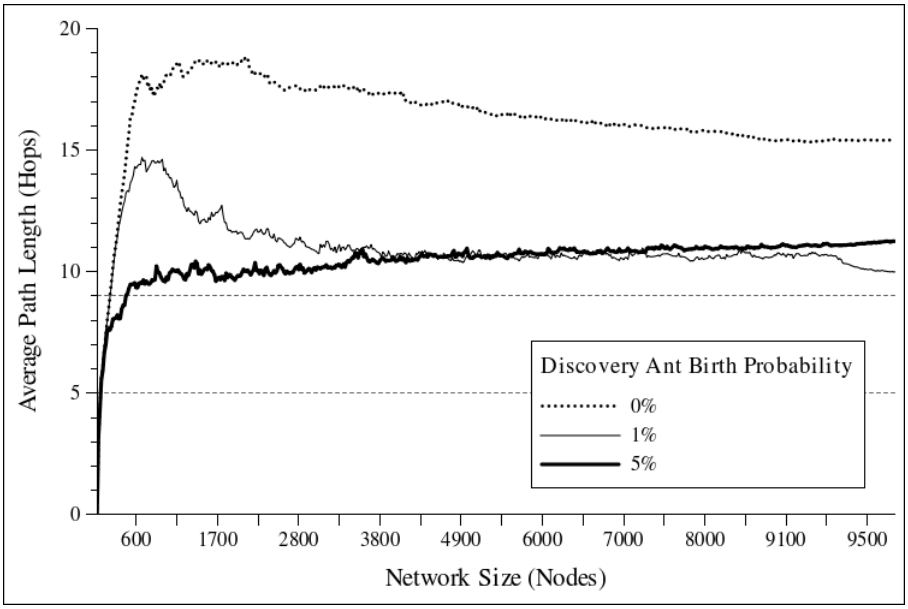

Figure 3. Average Path Length in Expanding Network

Group communication efficiency is assessed by determining the overall coverage (percentage of the nodes that received the message) as well as the total number of message forwardings. Every 25 iterations, messages are generated on 5 random nodes and multicasted on the network. As with resource discovery queries, we experimented with different strategies by varying the TTL and FW values.

For both resource discovery and group communication, we were also interested in evaluating the influence of network dynamics. In particular, for a fixed forwarding strategy with TTL=9 and FW=4, we experimented with progressively more stable networks by increasing connection/disconnection periods. Periods of 25, 50, 100, 250, 500, and 1000 iterations have been considered.

7. Results

Having detailed the parameters of the experimental evaluation scenarios, this section presents and discusses the obtained results. The first set of results mainly concerns the overlay management algorithm, whereas the second part concentrates on the efficiency of high level grid services such as resource discovery and group communication.

7.1 Scalability in an Expanding Network

Figure 3 shows the average path length in the expanding network scenario. Without optimizations, the construction protocol is able to keep an average length of 15 hops. In contrast, optimization tasks are able to keep up with the growth of the network, and bound path lengths in the overlay to values close to 10. Although a larger population of ants does not seem to improve the result, the number of edges in the resulting network (having an average of 9500 nodes) is significantly lowered: in average 46000 with 1%, 39000 with 5%, and 21000 without optimization.

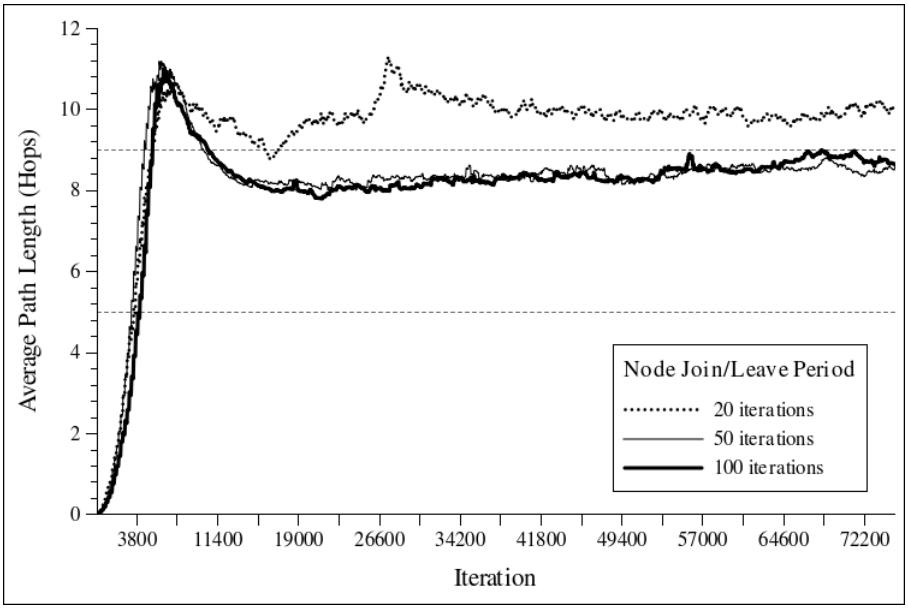

Figure 4. Reliability in Dynamic Scenarios

7.2 Reliability and Traffic Estimation in Dynamic Scenarios

Overlay reliability results aim at assessing the ability of the network to control dynamic conditions with high connection and disconnection rates. As shown in Figure 4, the algorithm is able to maintain an average path length close to 8.6 hops with rates of 50 and 100 iterations, and of 10 hops with 20 iterations. Although not shown in the figure, in all scenarios the network remains fully connected, confirming that the adopted recovery strategy is able to cope with abrupt disconnections.

After the initial construction phase, the average number of edges in the overlay is 6700. With a join-leave period of 50 iterations, according to packet size estimations provided in the previous section, at each iteration the algorithm generates a total average traffic of 20 kbytes, which results in a bandwidth consumption of about 3 bytes per link.

7.3 Comparison between **BlåtAnt-S** *and* **BlåtAnt-R**

Figure 5 illustrates how BlåtAnt-S obtains better results than BlåtAnt-R. In particular, the S version is able to better cope with the dynamics of the network, and maintains an average path close to $2D - 1 = 9$. The R version also generates more traffic, with about 25 kbytes per iteration: this difference is easily explained by the increased size of the *Discovery Ants* due to the additional neighbors' information.

7.4 Resource Discovery Efficiency

Resource discovery efficiency is evaluated both in respect of the success rate of single queries, as well as the traffic generated by the forwarding algorithm. Figure 6 shows the results of different forwarding strategies, with varying TTL and FW, concerning the average cost and success rate of one query. The histograms detail both the cost of maintaining the overlay (averaged for a rate of 5 queries every 25 iterations, for a total of 10000 queries) and the actual resource discovery query cost. Clearly, better result can be obtained at the expense of generating more traffic; it is nonetheless interesting to note how the difference between FW=6 and FW=8 is

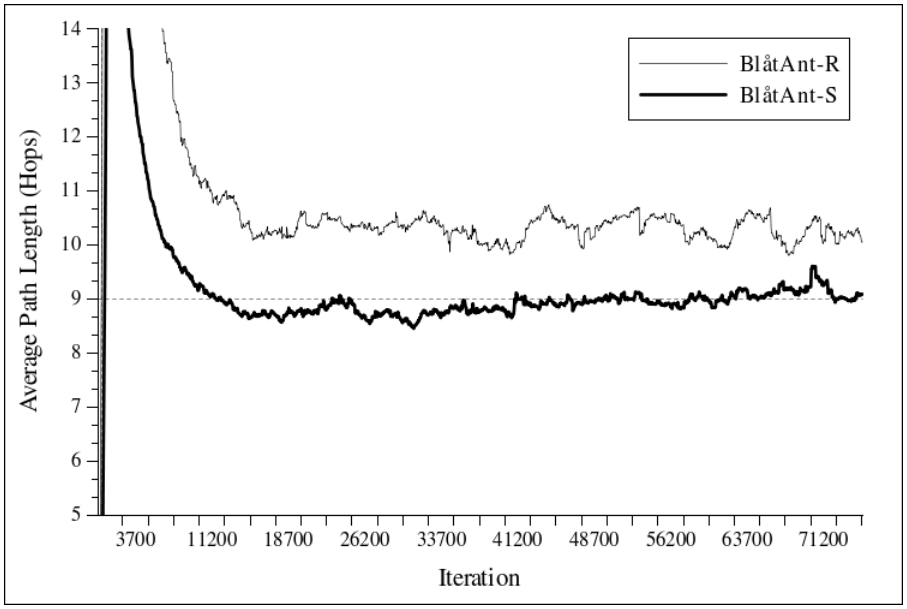

Figure 5. Average Path Length Comparison

limited: this is an outcome of the fact that most of the nodes have a degree less than 8. Results also show that the overlay management only counts for a negligible part of the traffic. A more detailed analysis of the results also showed that increasing the query traffic makes the overlay management even more efficient because less *Ping Ants* are needed.

Figure 7 shows group communication cost and performance. The best coverage (98% of an average of 1260 nodes) is obtained with a TTL of 9 and FW equal to 8, with an average of 5170 forwardings. It is interesting to note how the 9/6 scenario obtains similar coverage (98%) with slightly less forwardings: as for resource discovery, because the average number of neighbors per node is between 5 and 6, increasing the FW value does not provide significant advantages.

By taking into consideration the assumed size of queries and messages (1024 bytes), the results of the different forwarding strategies used by resource discovery and group communication become clear; respectively, queries are stopped as soon as a match is found, whereas messages are not. For the same TTL and FW parameters values, resource discovery thus produces in average 200 kbytes of traffic less than group communication.

7.5 Network Dynamics Influence

As illustrated in Figures 8 and 9, both resource discovery and group communication efficiency are affected by the rate of connections and disconnections in the overlay. As expected, the resource discovery hit rate as well as the group communication coverage increase in more stable scenarios. On the other side, although the overlay management related traffic decreases with larger connection/disconnection periods (scenarios of 50 and 100 iterations), the overall traffic increases: as the network managing algorithm is able to further optimize the average path length, more nodes are contacted within the 9 hops radius, thus more messages are transmitted. Nonetheless, it is interesting to note that in even more stable networks (250, 500 and 1000 iterations) the optimization process is more effective in that it has time to remove additional redundant links, resulting in less bandwidth being consumed.

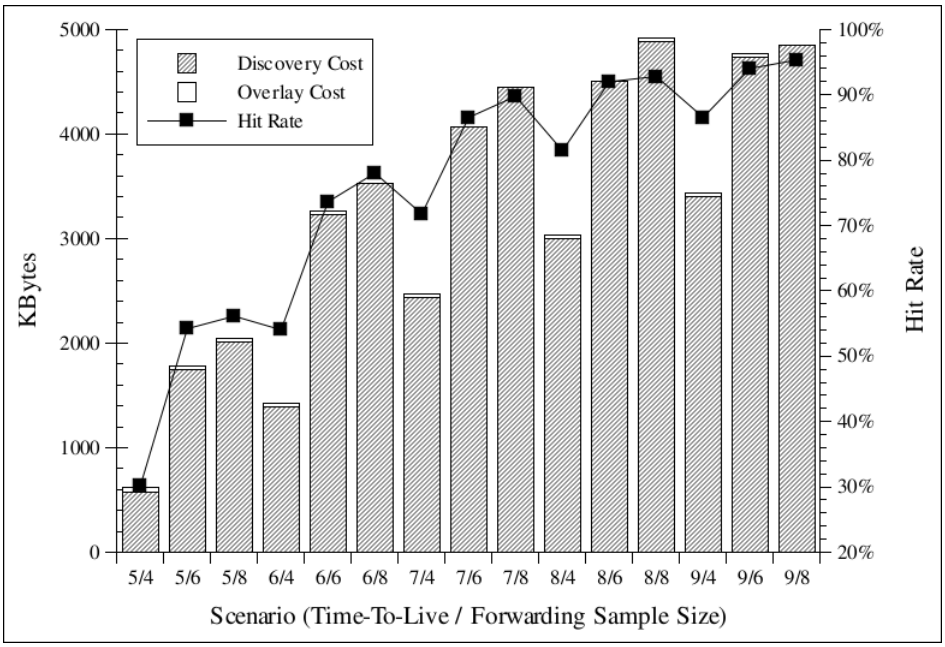

Figure 6. Resource Discovery Efficiency

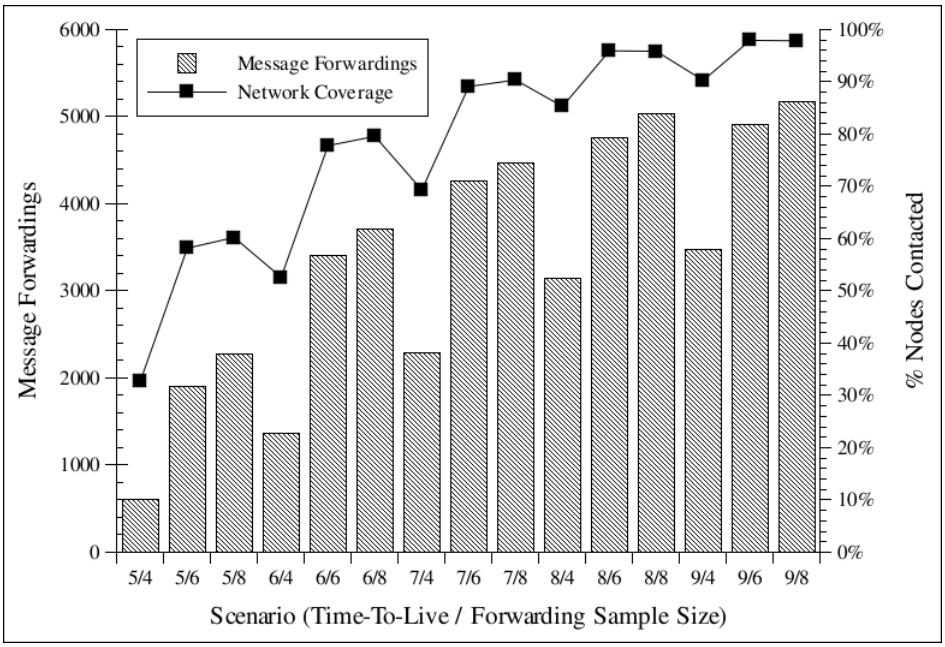

Figure 7. Group Communication Efficiency

8. Conclusions

A fundamental milestone in the path towards next-generation grids is the deployment of flexible, autonomic, and self-manageable solutions for efficient, scalable and robust grid service provisioning. In accordance with this vision, we introduced a framework for grid service provisioning composed of two functional layers: a self-organized peer-to-peer overlay management layer, and a grid services interface layer. The proposed architecture promotes a clear separation of concerns: network communication and membership are managed at the lower level, whereas grid services are provided by the higher level.

The major novelty of this work is introduced within the overlay management

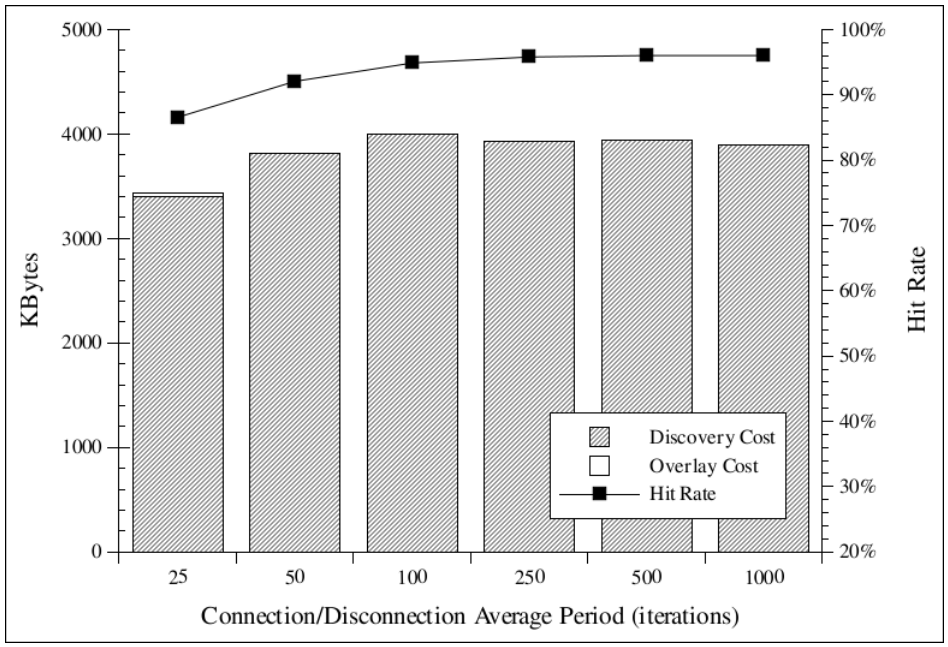

Figure 8. Network Dynamics Influence on Resource Discovery (9/4 Scenario)

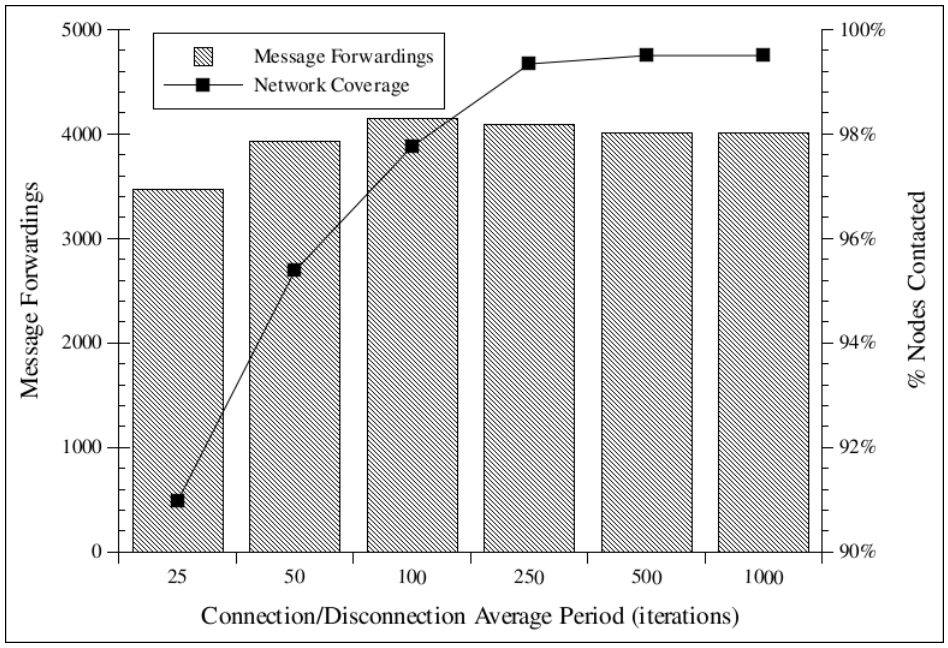

Figure 9. Network Dynamics Influence on Group Communication (9/4 Scenario)

layer, which relies on a collaborative bio-inspired algorithm to maintain an optimized topology and constitutes an efficient communication platform. Different types of mobile software agents, whose behavior is inspired by insect colonies, collaborate in a fully distributed way to bound the distance between each pair of nodes, by means of creating additional links and rearranging existing ones. The adopted solution provides an adaptive, reliable and robust communication platform to connect grid nodes and ensure efficient operation of the grid.

The upper service provisioning layer hides the complexity of the network by providing grid applications with an interface to the resources shared on the grid. In particular, the layer delivers grid resource discovery and monitoring, as well as

best-effort multicast group communication.

Different evaluation scenarios were considered and experiments demonstrated the qualities of the framework. In particular, the adaptiveness, scalability and fault resiliency of the overlay management was proved. The constructed overlay is able to cope with dynamic networks and faulty nodes, as well as to optimally reorganize connections in order to bound the average path length. Furthermore, simulations of both the resource discovery and the group communication services demonstrated the suitability of our solution for solving grid tasks.

Future work focuses on the integration of additional techniques to improve the efficiency of the resource discovery process [56], as well as the inclusion of this framework within the overall SmartGrid [49] platform.

9. Acknowledgments

The authors would like to thank Dr. Apostolos Malatras and Fulvio Frapolli for their useful feedback. This research is financially supported by the Swiss Hasler Foundation in the framework of the "ManCom Initiative", project Nr. 2122.

References

[1] M. Parashar and C. Lee. Grid computing - introduction and overview. *Proceedings of the IEEE: Special Issue on Grid Computing*, 93(3), March 2005.

[2] A. Andrzejak, A. Reinefeld, F. Schintke, T. Schtt, C. Mastroianni, P. Fragopoulou, D. Kondo, P. Malecot, G. Cosmin Silaghi, L. Moura Silva, P. Trunfio, D. Zeinalipour-Yazti, and E. Zimeo. Grid architectural issues: State-of-the-art and future trends, May 2008.

[3] Ian T. Foster. Globus toolkit version 4: Software for service-oriented systems. In Hai Jin, Daniel A. Reed, and Wenbin Jiang, editors, *NPC*, volume 3779 of *Lecture Notes in Computer Science*, pages 2–13. Springer, 2005.

[4] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience*, 17(2-4):323–356, 2005.

[5] R. Buyya, D. Abramson, and J. Giddy. Nimrod/g: An architecture for a resource management and scheduling system in a global computational grid. In *HPC ASIA2000*. IEEE, 2000.

[6] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on*, pages 181–194, 2001.

[7] Xuehai Zhang, Jeffrey L. Freschl, and Jennifer M. Schopf. A performance study of monitoring and information services for distributed systems. In *HPDC '03: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, page 270, Washington, DC, USA, 2003. IEEE Computer Society.

[8] Carlo Mastroianni, Domenico Talia, and Oreste Verta. Designing an information system for grids: Comparing hierarchical, decentralized p2p and super-peer models. *Parallel Comput.*, 34(10):593–611, 2008.

[9] Shishir Bharathi Ann Chervenak. Peer-to-peer approaches to grid resource discovery. In *Making Grids Work*, pages 59–76, 2008.

[10] R. Ranjan, A. Harwood, and R. Buyya. Peer-to-peer-based resource discovery in global grids: a tutorial. *Communications Surveys & Tutorials, IEEE*, 10(2):6–33, 2008.

[11] Adriana Iamnitchi and Ian Foster. A peer-to-peer approach to resource location in grid environments. In *Grid resource management: state of the art and future trends*, pages 413–429, Norwell, MA, USA, 2004. Kluwer Academic Publishers.

[12] Diego Puppin, Stefano Moncelli, Ranieri Baraglia, Nicola Tonellotto, and Fabrizio Silvestri. A grid information service based on peer-to-peer. In *Euro-Par 2005 Parallel Processing*, pages 454–464, 2005.

[13] I. Stoica, R. Morris, D. Karger, F. M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01*, volume 31, pages 149–160, New York, NY, USA, October 2001. ACM Press.

[14] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218, 2001.

[15] C. Schmidt and M. Parashar. Flexible information discovery in decentralized distributed systems. *12th IEEE International Symposium on High Performance Distributed Computing*, June 2003.

[16] Sameh El-Ansary, Per Brand, and et al. Haridi Seif. Efficient broadcast in structured p2p networks. In *Proceedings of 2nd International Workshop On Peer-To-Peer Systems (IPTPS'03)*, Berkeley, CA, USA, 2003.

[17] Domenico Talia, Paolo Trunfio, Jingdi Zeng, and Mikael Hgqvist. A dht-based peer-to-peer framework for resource discovery in grids. Technical Report TR-0048, Institute on System Architecture, CoreGRID - Network of Excellence, June 2006.

[18] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a dht. In *ATEC '04*, Berkeley, CA, USA, 2004. USENIX Association.
[19] Haiying Shen. A p2p-based intelligent resource discovery mechanism in internet-based distributed systems. *J. Parallel Distrib. Comput.*, 69(2):197–209, 2009.
[20] Ghamri-Doudane, Samir, Agoulmine, and Nazim. Enhanced dht-based p2p architecture for effective resource discovery and management. *Journal of Network and Systems Management*, 15(3):335–354, September 2007.
[21] M. Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *Peer-to-Peer Computing, 2001.*, August 2001.
[22] I. Clarke, O. Sandberg, b. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009, 2001.
[23] B. Beverly Yang and H. Garcia-Molina. Designing a super-peer network. pages 49–60, March 2003.
[24] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *SIGMETRICS '02: Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 258–259, New York, NY, USA, 2002. ACM.
[25] A. Forestiero, C. Mastroianni, and G. Spezzano. Building a peer-to-peer information system in grids via self-organizing agents. *Journal of Grid Computing*, 2008.
[26] Vassilios V. Member-Dimakopoulos and Evaggelia Member-Pitoura. On the performance of flooding-based resource discovery. *IEEE Trans. Parallel Distrib. Syst.*, 17(11):1242–1252, 2006.
[27] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. In *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on*, pages 5–14, 2002.
[28] C. Gkantsidis, M. Mihail, and A. Saberi. Hybrid search schemes for unstructured peer-to-peer networks. volume 3, pages 1526–1537 vol. 3, March 2005.
[29] A. Iamnitchi and I. Foster. On fully decentralized resource discovery in grid environments. In *Int. Workshop on Grid Computing*, 2001.
[30] Elena Meshkova, Janne Riihijärvi, Marina Petrova, and Petri Mähönen. A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks. *Comput. Netw.*, 52(11):2097–2128, 2008.
[31] Adriana Iamnitchi and Ian T. Foster. On fully decentralized resource discovery in grid environments. In *GRID '01: Proceedings of the Second International Workshop on Grid Computing*, pages 51–62, London, UK, 2001. Springer-Verlag.
[32] Vana Kalogeraki, Dimitrios Gunopulos, and D. Zeinalipour-Yazti. A local search mechanism for peer-to-peer networks. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 300–307, New York, NY, USA, 2002. ACM.
[33] Christos Gkantsidis, Milena Mihail, and Amin Saberi. Random walks in peer-to-peer networks: algorithms and evaluation. *Perform. Eval.*, 63(3):241–263, 2006.
[34] Xiuqi Li and Jie Wu. A hybrid searching scheme in unstructured p2p networks. *Int. J. Parallel Emerg. Distrib. Syst.*, 22(1):15–38, 2007.
[35] Edith Cohen and Scott Shenker. Replication strategies in unstructured peer-to-peer networks. *SIGCOMM Comput. Commun. Rev.*, 32(4):177–190, 2002.
[36] Agostino Forestiero, Carlo Mastroianni, Giuseppe Papuzzo, and Giandomenico Spezzano. Towards a self-structured grid: An ant-inspired p2p algorithm. *Transactions on Computational Systems Biology X*, pages 1–19, 2008.
[37] Vicent Cholvi, Pascal Felber, and Ernst Biersack. Efficient search in unstructured peer-to-peer networks. In *SPAA '04: Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 271–272, New York, NY, USA, 2004. ACM.
[38] Arturo Crespo and Hector Garcia-Molina. Routing indices for peer-to-peer systems. In *ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 23, Washington, DC, USA, 2002. IEEE Computer Society.
[39] J. Kleinberg. Complex networks and decentralized search algorithms. In *Proceedings of the International Congress of Mathematicians (ICM)*, 2006.
[40] D. Stutzbach, R. Rejaie, and S. Sen. Characterizing unstructured overlay topologies in modern p2p file-sharing systems. *Networking, IEEE/ACM Transactions on*, 16(2):267–280, April 2008.
[41] Robert A. Ghanea-Hercock, Fang Wang, and Yaoru Sun. Self-organizing and adaptive peer-to-peer network. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36(6):1230–1236, Dec. 2006.
[42] Young June Pyun and Douglas S. Reeves. Constructing a balanced, (log(n)/loglog(n))-diameter super-peer topology for scalable p2p systems. In *P2P '04: Proceedings of the Fourth International Conference on Peer-to-Peer Computing*, pages 210–218, Washington, DC, USA, 2004. IEEE Computer Society.
[43] Vincenza Carchiolo, Michele Malgeri, Giuseppe Mangioni, and Vincenzo Nicosia. Fast information retrieval in a self-organising p2p network. *Journal of Computers (JCP)*, 2(5):75–81, 2007.
[44] Karl Aberer, Philippe Cudré-Mauroux, Anwitaman Datta, Zoran Despotovic, Manfred Hauswirth, Magdalena Punceva, and Roman Schmidt. P-grid: a self-organizing structured p2p system. *SIGMOD Rec.*, 32(3):29–33, 2003.
[45] M. Ripeanu, A. Iamnitchi, I. Foster, and A. Rogers. In search of simplicity: A self-organizing group communication overlay. *SASO '07*, pages 371–374, July 2007.
[46] K. Shen. Structure management for scalable overlay service construction. In *NSDI'04*, pages 21–21. USENIX Association, 2004.
[47] M. Amoretti, F. Zanichelli, and G. Conte. Sp2a: a service-oriented framework for p2p-based grids. In *MGC '05*, pages 1–6, New York, NY, USA, 2005. ACM.
[48] Amos Brocco and Béat Hirsbrunner. Service provisioning framework for a self-organized grid. ICCCN 09 Workshop on Grid and P2P Systems and Applications (GridPeer 2009), IEEE, August 2009.
[49] Y. Huang, A. Brocco, B. Hirsbrunner, M. Courant, and P. Kuonen. Smartgrid: A fully decentralized grid scheduling framework supported by swarm intelligence. In *7th Int. Conf. on Grid and Cooperative*

Computing. GCC2008, October 2008.

[50] Amos Brocco, Fulvio Frapolli, and Béat Hirsbrunner. Bounded diameter overlay construction: A self organized approach. In *IEEE Swarm Intelligence Symposium*. SIS 2009, IEEE, April 2009.

[51] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, Inc., New York, NY, USA, 1999.

[52] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[53] D. Fellows A. Ly S. McGough D. Pulsipher A. Anjomshoaa, M. Drescher and A. Savva. Job submission description language (jsdl) specification, version 1.0. Technical report, Global Grid Forum.

[54] Sergio Andreozzi, Stephen Burke, Felix Ehm, Laurence Field, Gerson Galang, Balazs Konya, Maarten Litmaath, Paul Millar, and J. P. Navarro. Glue specification v. 2.0. Technical report, Open Grid Forum, March 2009.

[55] ZhenChun Huang, Lei Gu, Bin Du, and Chuan He. Grid resource specification language based on xml and its usage in resource registry meta-service. pages 467–470, Sept. 2004.

[56] Amos Brocco, Apostolos Malatras, and Béat Hirsbrunner. Proactive information caching for efficient resource discovery in a self-structured grid. In *Workshop on Bio-Inspired Algorithms for Distributed Systems*. ICAC 2009, ACM, June 2009.