Egyptian Informatics Journal (2011) 12, 125–138

Cairo University

Egyptian Informatics Journal

www.elsevier.com/locate/eij
www.sciencedirect.com

ORIGINAL ARTICLE

PSG: Peer-to-Peer semantic grid framework architecture

Amira Soliman [a,*], Amr Kamel [b], Walaa Sheta [a], Reem Bahgat [b]

[a] Informatics Research Institute, City for Scientific Research and Technology Applications (MuCSAT), Alexandria 21934, Egypt
[b] Computer Science Department, Faculty of Computers and Information, Cairo University, Giza 12613, Egypt

Received 1 March 2011; revised 12 May 2011; accepted 5 June 2011
Available online 5 July 2011

KEYWORDS

Semantic grid;
Peer-to-Peer systems;
Multi-agent systems;
Semantic resource discovery;
Ontology

Abstract The grid vision, of sharing diverse resources in a flexible, coordinated and secure manner, strongly depends on metadata. Currently, grid metadata is generated and used in an ad-hoc fashion, much of it buried in the grid middleware code libraries and database schemas. This ad-hoc expression and use of metadata causes chronic dependency on human intervention during the operation of grid machinery. Therefore, the Semantic Grid is emerged as an extension of the grid in which rich resource metadata is exposed and handled explicitly, and shared and managed via grid protocols. The layering of an explicit semantic infrastructure over the grid infrastructure potentially leads to increase interoperability and flexibility. In this paper, we present PSG framework architecture that offers semantic-based grid services. PSG architecture allows the explicit use of semantics and defining the associated grid services. PSG architecture is originated from the integration of Peer-to-Peer (P2P) computing with semantics and agents. Ontologies are used in annotating each grid component, developing users/nodes profiles and organizing framework agents. While, P2P is responsible for organizing and coordinating the grid nodes and resources.

1. Introduction

Computational grids are sharing environments in which collections of geographically distributed hardware and software resources are made available to groups of remote users. Ian Foster et al. [1] define the grid problem as coordinated resource sharing and problem solving in dynamic and multi-institutional virtual organizations. Grid computing combines the technologies of distributed computing, networks, servers and storages, and supports resource sharing across different organizations. After almost 20-year development, grid computing has many varieties, such as computational grid, data grid, access grid, information grid, service grid, wireless grid and cloud computing [2].

A main factor that drives the development and evolution of the grid is the necessity to face the enormous amount of data

* Corresponding author.
E-mail addresses: a.soliman@mucsat.sci.eg (A. Soliman), a.kamel@fci-cu.edu.eg (A. Kamel), wsheta@mucsat.sci.eg (W. Sheta), r.bahgat@fci-cu.edu.eg (R. Bahgat).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.
doi:10.1016/j.eij.2011.06.001

Production and hosting by Elsevier

that any field of human activity is producing. Although databases and data warehouses are nowadays more powerful and can manage large datasets, only a small portion of data will be accessed by humans or programs. The obstacle is not the technology to store and to access data, but perhaps what is lacking is the ability to transform data tombs in useful data and extract knowledge from them [3]. Therefore, the grid is moving from computation and data management, to a pervasive world-wide knowledge management infrastructure. To achieve this very ambitious goal, Next Generation Grid is introduced to develop grid frameworks based on the key technologies available such as Semantic Web, Peer-to-Peer Systems, and Ambient Computing [4].

The Semantic Grid is an initiative of the UK EPSRC/DTI Core e-Science Program [5] that aims to integrate and bridge the efforts made in the grid and in the Semantic Web [6]. The Semantic Grid is an extension of the current grid in which information and services are given well-defined meaning, better enabling computers and people to work in cooperation. This approach is essential to achieve the full richness of the grid vision, with a high degree of easy-to-use and seamless automation, and hence, enables flexible collaborations and computations on a global scale. Through provision of ontological support to the grid, there is the potential to create a searchable, reusable resource that is understandable by and accessible to a wider community [4].

Currently, grid metadata is generated and used in an ad-hoc fashion, much of it buried in the grid middleware code libraries and database schemas. This ad-hoc expression and use of metadata causes chronic dependency on human intervention during the operation of grid machinery, which also leads to systems that are brittle when faced with frequent syntactic changes in resource coordination and sharing protocols. So, in this paper, we introduce the architecture of a lightweight framework called PSG that offers semantic-based grid services in open and distributed environments. PSG architecture allows the explicit use of semantics and defining the associated grid services to support a variety of service capabilities. We develop grid ontology and use it in annotating each grid component. Furthermore, the developed ontology is used to build nodes profiles, organize framework agents, and manage agent communication.

The PSG architecture aims to provide pure ad-hoc grid with ontology-based semantic modeling of users tasks/needs, grid services and data sources. In this architecture, we use P2P and multi-agent models to achieve self-configuration, autonomic management, dynamic resource discovery and fault-tolerance. The introduced architecture is fully decentralized and is able to operate in open environments without using of pre-existing infrastructures or central administration. Moreover, in our design, we integrate ontology with the grid services in order to achieve effective reuse of grid information, intelligent searching and improve interlinking among different grid resources. We have developed a prototype implementation of the PSG architecture to obtain experimental results on constructing P2P overlay and providing semantic services. In this prototype, we integrate the grid framework ontology with the ontology of social network application.

The paper is organized as follows. Section 2 presents some related work on semantic grids including some attempts towards integrating ontologies. Section 3 introduces the main PSG components and services. In Section 4, the detailed framework architecture is presented. Section 5 introduces the details of developed PSG social network application. Section 6 lists the experimental results we obtain. Finally, Section 7 concludes this paper and some work that can be investigated in the future.

2. Related work

The Open Grid Services Architecture (OGSA) [7] is the result of a standardization effort, and now it is sustained by the grid standards body, namely the Open Grid Forum (OGF) [8]. OGSA aims to define a core set of capabilities and behaviors for grid systems. Semantic-OGSA (S-OGSA) [9] is a reference architecture that extends OGSA to support the explicit handling of semantics, and defines the associated knowledge services to support a spectrum of service capabilities. The objective of S-OGSA is the provision of a unified platform for exposing and delivering explicit metadata in grid applications, including a formal framework and a set of guidelines to ease the development of semantic grid applications. S-OGSA has three main aspects: the model (the elements that it is composed of and its interrelationships), the capabilities (the services needed to deal with such components) and the mechanisms (the elements that will enable communication when deploying the architecture in an application) [9].

Another architecture, that provides distributed data management based on semantics, is Open Grid Services Architecture Data Access and Integration (OGSA-DAI) [10]. OGDA-DAI is about sharing data, whether this data lies within a single organization, between a group of partners, or with the public. By sharing data, OGSA-DAI can identify, understand and exploit complex interactions between disparate variables and so convert data into information. Besides, OGSA-DAI allows data in distributed databases to be accessed, updated, transformed and combined. OGSA-DAI has a powerful distributed query processor that allows queries to be run over many databases as if they were a single database. OGSA-DAI can be used with relational and XML databases and with file systems [10].

Overall, there is still the need for lightweight semantic grids architectures that support applications running on purely ad-hoc networks. Crucially, such grid applications need a completely decentralized and collaborative approach to the resource discovery and coordination. Although, OGSA-DAI and S-OGSA allow grid services to be exploit semantics, the installation and deployment of applications on them require dedicated pre-existing configured core servers. Hence, there still the need to have a lightweight architecture enables users to form spontaneous semantic service networks with intelligent searching and improved interlinking without using of pre-existing infrastructures or central administration.

3. Main framework components

PSG architecture comprises three main components: P2P overlay, Multi-Agent System (MAS) hierarchy, and Semantics. P2P overlay is used to build non-hierarchical decentralized grid services and hence increase grid scalability. Furthermore, using P2P model provides a fully decentralized system without using of pre-existing infrastructure or central administration. Semantic services are responsible for integrating ontologies with grid entities and providing ontology matchmaking

mechanisms for resource discovery. Moreover, semantic services provide users with intelligent reasoning and interlinking of available resources on the grid. The reason behind using Multi-agent systems in our framework is to provide both P2P and semantic services with flexible and decentralized decision making capabilities.

3.1. Peer-to-Peer overlay

Recently, Peer-to-Peer (P2P) systems have gained tremendous popularity. P2P systems consist of a dynamically changing set of nodes with symmetric roles connected via the Internet. P2P is a class of self-organizing systems or applications that takes advantage of distributed resources storage, processing, information, and human presence [11,12]. P2P systems have emerged as a general paradigm for constructing resilient, large-scale, and distributed services and applications in the Internet. Therefore, P2P and grid aim to provide access to remote computing resources for high-performance, data-intensive applications. However, most of today's grid frameworks are developed as centralized or hierarchical architectures. Hence, as grid sizes increase, these architectures suffer from bottlenecks and scalability problems [4]. While, adopting P2P models in grid architectures present good scalability, efficiency, flexibility, and robustness for grid computing in open environments [11].

P2P overlay networks for the Internet have been classified based on the structure of constructed network overlay into two categories: unstructured P2P overlay networks and structured P2P overlay networks [13,14]. In unstructured systems such as Gnutella [15], the placement of data files is totally isolated from the overlay topology and random search algorithms are used to locate data files. In the contrary, structured systems define the relationships between nodes and data files. Numerous structured P2P overlays have been proposed, such as CAN [16], Chord [17], and Pastry [18]. In our framework, nodes are organized using Chord protocol. The Chord maps the key onto a node using consistent hashing function. Moreover, The Chord protocol specifies how to find the locations of keys, how new nodes join the system, and how to recover from the failures of existing nodes [17].

Chord provides consistent hashing to generate nodes and keys identifiers. The node identifiers are arranged in a circle that is called the Chord ring. Every node keeps a table called the finger table which stories node neighbors' identifiers. In an N-node network, each node's routing table stores information only about O(log N) other nodes. Every key k is assigned to the first node whose identifier n is equal to or larger than k. This node is called the successor node of key k. If node n does not know the successor of a key k, n can find a node m in its finger table whose identifier is closer than its own to k. Then, node m will know more about the identifier circle in the region of k than n does. By repeating this process, n learns about nodes with identifiers closer and closer to k [17]. Fig. 1 shows an identifier circle with three nodes (0, 3, and 6) and three keys (keys 1 and 2 are located at node 3, key 5 at node 6). Also, nodes finger tables are shown.

3.2. Multi-agent system

Agent-based systems technology has generated lots of excitement in recent years because of its promise as a new paradigm for conceptualizing, designing, and implementing software systems in open and dynamic environments. Agent architectures are designed to exhibit autonomy, decentralized coordination, and complex distributed behaviors in highly dynamic environments such as grids [19,20]. Currently, agents on the internet mostly perform information retrieval and filtering. When agents and grids systems cooperate together, agents will perform information gathering in context and sophisticated reasoning in support of user problem-solving tasks. Some of the gained benefits of this integration are modularity, autonomy, swarm-level coordination, and immersion in their environment, to more advanced capabilities such as problem solving, meta-reasoning, learning, shared goals, and human system collaboration [21].

Agent would not be considered as intelligent software models if they repeated the same mistakes and never improved their performance in tasks they perform routinely. Barbara Hayes-Roth [22] defines the primary objective of an intelligent agent as "to maintain the value of its own behavior within an acceptable

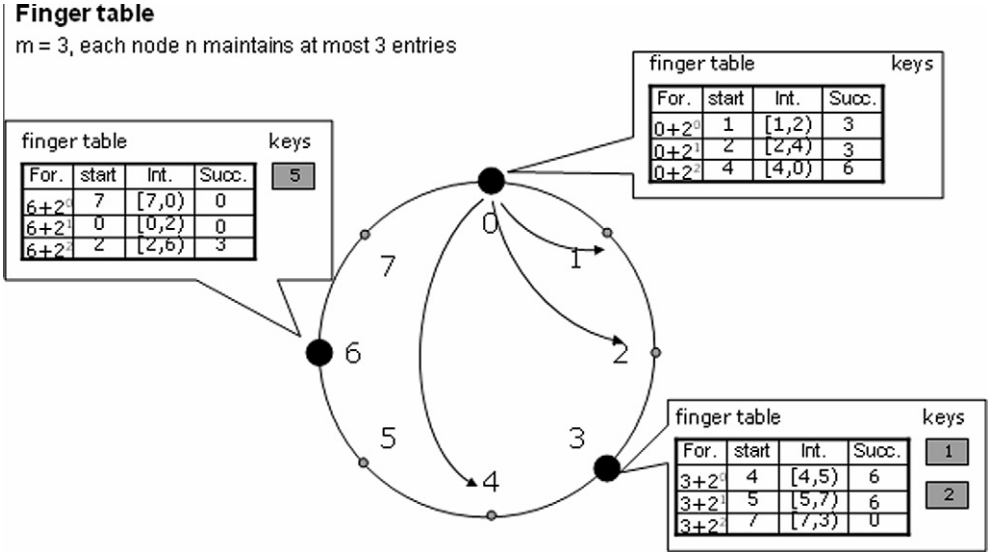

Figure 1 An identifier circle consisting of the three nodes 0, 3, and 6.

range over time''. From these requirements, there are two important properties for an intelligent agent, flexibility (the agent should react to important unexpected events) and timeliness (the agent should meet various real-time constraints) [22]. Learning can also improve agent performance by recognizing common failure conditions and designing mechanisms to anticipate and avoid them. Also, Meta-reasoning enables agent to make decisions about what to reason about, setting its own goals and deciding how to allocate its resources [21].

The BDI (belief-desire-intention) model is well understood as an agent architecture to support goal oriented behaviour in intelligent agents. It is a theory of practical reasoning that represents an abstraction of human deliberation based on a theory of rational actions in the human cognition process. The mental attitudes of belief, desire and intention represent the information, motivational, and deliberative states of the agent respectively. To process a certain matter, a BDI agent goes through three phases as follows: firstly, perceiving the environment; secondly deliberating on what to do and how to do it and finally executing the action plans. Moreover, the process of reconsidering agent desires and intentions is important to provide system with flexible and adaptive decisions [23,24]. Fig. 2 shows BDI agent architecture.

The basic agent control loop of the BDI interpreter consists of perception, updating belief, generating desires, choosing intention and executing actions. Desires, intentions, and then actions are generated based on beliefs. In our framework implementation we build an ontology using Protégé tool [25] in order to handle agents BDI components such as beliefs, goals, and execution plans. We build DBI ontology so that beliefs can be updated and intentions could be reconsidered during agent task execution. Section 4.2, Agents Hierarchy, illustrates the details of agent BDI concepts in developed framework ontology.

3.3. Semantic model

Resource discovery in grid is about finding relevant resources, the overall quality of a discovery service is determined not only by usual quality of service measures such as performance, reliability and availability, but also by its accuracy that measures how many of the discovered resources are relevant, and how relevant they are. Accurate resource discovery should be able to find the best approximate matches for the user. Hence, resource discovery in grid has to deal with large number of volatile resources described using different approaches and languages, and managed by different virtual organizations. In such heterogeneous and dynamic environments, syntactic keyword and taxonomy-based matching is insufficient to achieve high precision resource discovery. In order to improve the precision of a discovery service, resources must be given well-defined meaning carried by semantic information added to resource descriptions [19,26,27].

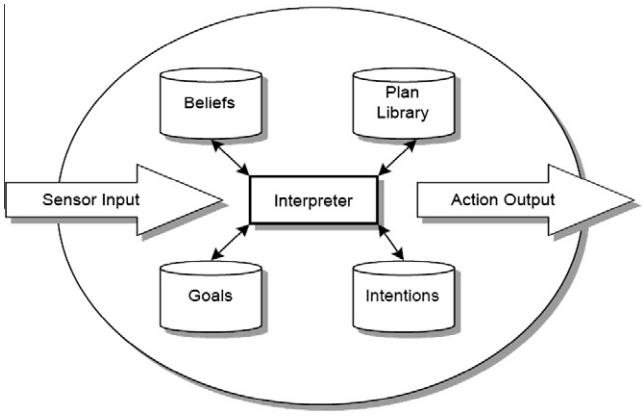

Figure 2 Agents and BDI process.

Through provision of ontological support to the grid, there is the potential to create a searchable, reusable resource that is understandable by and accessible to a wider community. This requires that all kind of grid content to be marked up with metadata that encodes its meaning in a way that is machine-interpretable and hence be processed by agents, search engines and applications to automate the content discovery [6,27]. Ontology represents the vocabulary terms, and how they inter-relate, for the concepts shared by a community [6,11]. It formally specifies how to represent objects, concepts and other entities that are assumed to exist in some area of interest and the relationships among them. Thus, ontologies are used for constituting a community reference, sharing consistent understanding of what information means, making possible knowledge reuse and sharing, and increasing interoperability between systems [28,29].

Ontologies are used in Artificial Intelligence, Semantic Web, Software Engineering and Information Science as a form of knowledge representation about the world. Most ontologies describe individuals or instances (ground level components of ontology; they may include concrete objects such as people, animals, and planets, as well as abstract individuals such as numbers and words), classes or concepts (abstract groups, sets, or collections of objects), attributes, and relations. Objects in the ontology can be described by assigning attributes to them. Each attribute has at least a name and a value, and is used to store information that is specific to the object it is attached to. For example, the Person object has attribute named Age with value 20. An important use of attributes is to describe the relationships (also known as relations) between objects in the ontology. Typically, a relation is an attribute whose value is another object in the ontology [30,31].

Our proposed framework, PSG, is developed using Service Oriented Architecture (SOA) that integrates ontology with services definitions. PSG provides conceptual model to describe resources and services. We develop framework ontology that describes every service and entity in the PSG architecture. Moreover, PSG conceptual model is used to increase mechanisms of the activities related to services, namely discovery and negotiation. PSG framework ontology is developed according to OWL-DL [33] ontology language specifications and describes peers, agents, tasks, and resources. Moreover, framework ontology creates agent hierarchy and manages agent communication based on BDI reasoning model. Task execution is also included in framework ontology; each task is mapped to a set of related actions to be executed and alternatives plans could be used in case of failure of main plan.

4. The proposed framework

4.1. Layered architecture

The framework functionalities are distributed on three layers where each layer has its own responsibilities. The layers as

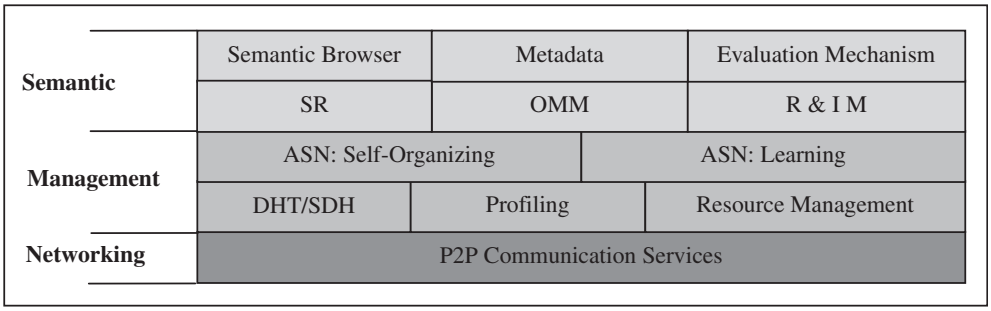

Figure 3 Framework layered architecture.

shown in Fig. 3 and they are Networking, Management, and Semantic. Networking layer is responsible for presence management, organizing nodes in Chord ring, and finally distributing resources keys. Management layer is responsible for organizing agents into Agent Semantic Network (ASN) according to their roles. The framework uses two ontologies; we develop the first and integrate it with agents while the second ontology covers application domain and by which grid resources are indexed and organized. Finally, the semantic layer that handles the ontological support that is either provided to grid users or lower layer agents.

The networking layer provides the service of creating spontaneous services network that connects the participating nodes. As we mention before, the nodes are organized in P2P overlay using Chord protocol [17]. We use the Chord implementation of METEOR project [41]. METEOR is an open source project offered by JXTA community [32]. This layer contains the implementation of two services: chord service and routing service. The chord service is implemented using JXTA Discovery and Resolver protocols [33]. While, the routing service is the intermediate between chord service and network routing protocol. The routing service stores the routing paths among the different participating nodes. The path is stored from the requesting (i.e., source) node to the destination node.

Management layer consists of the main services provided by PSG starting from Resource Management service that organizes resources using Semantic Driven Hashing (SDH), and the Profiling service that keeps up-to-date profiles of both users and nodes. Finally, management layer provides agent services, each agent role is provided as a separate service. Agent Management Service (AMS) is responsible for managing the communication between agent services and reporting environmental changes to update agent model.

The role of semantic layer is to provide the ontological support to users and middleware agents. A set of facilities is provided to users to enable them to interact with the framework like browsing, providing metadata, and finally evaluation mechanisms. Browsing facility helps users to browse and navigate through available resources and used ontologies concepts. While, evaluation mechanisms affect the decisions taken by ontology agents in matchmaking process. Another set of facilities is developed to guide ontology agents starting from handling rules for Semantic Relationships (SR) among different ontology concepts, then, Ontology Matchmaking (OMM) process that is based on classification matching scheme, and finally reasoning and integration mechanisms (R&IM) using Rules (i.e., Inference Rules).

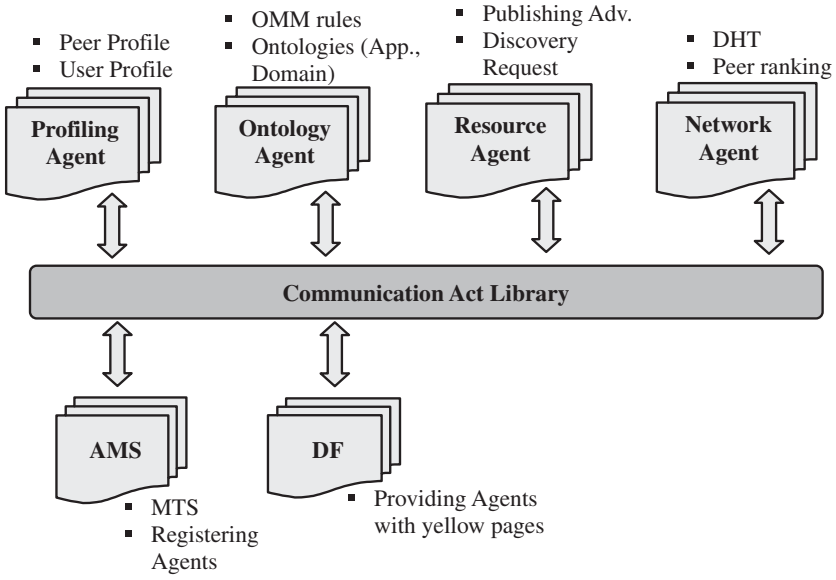

Figure 4 Different agent roles.

4.2. Agent hierarchy

Since agents are autonomous entities, coordination is not a passive task, but involves actively agents themselves, in other words, agents collaborate in order to successfully coordinate themselves. There are several approaches related to agent collaboration and coordination, Roles is one of them [42]. In our framework agent can play a role from six available roles that are: Agent Management (AM), Directory Facilitator (DF), Network Management role (NM), Resource Management role (RM), Ontology Management role (OM), and finally Profiling Management role (PM). Fig. 4 shows these roles and different role's responsibilities.

Agents with NM role are responsible for joining the chord ring, distributing resources keys, and finally managing peer routing table. While, Agents with PM role are responsible for managing users profiles and resources accessing polices. Also, providing up-to-date peer status and storing the changes in peer profile. RM role covers resources advertising and discovery. So, RM agents handle publishing resources advertisements between peers and respond to discovery queries either by sending requested resource if it is its owner, or sending routing information of peer holding resource. OM agents are responsible of handling all issues related to domain ontology. OM agents responsibility starts from loading the ontology files, then manages relationships between domain concepts. OM agent is responsible for assembling semantic relationships and constructing concepts subsumption hierarchy from the domain ontology. Subsequently, OM agent provides the SDH and matchmaking services for RM agents.

To allow agents to interact with each other, we provide a communication act library that is based on FIPA specifications [43]. This communication act library deals with constructing agent messages and transporting these messages where the message represents one of the agent communicative acts. AM has a main role in providing the communication functionality for other agents as we design it as the communicator to eliminate the communication overhead from other agents. Here we list the communicative acts that we use in our framework:

- Request: The sender requests the receiver to perform some action.
- Agree: The action of agreeing to perform some action, possibly in the future.
- Refuse: The action of refusing to perform a given action, and explaining the reason for the refusal.
- Cancel: The action of first agent informing another agent that it is no longer requires that the second agent performs some action.
- Failure: The action of telling another agent that an action was attempted but the attempt failed.
- Inform: The sender informs the receiver that a given proposition is true.
- QueryIf: The action of asking another agent whether or not a given proposition is true.

In our framework implementation we build an ontology using Protégé tool [25], PSG ontology covers BDI concepts, and execution plans. Beliefs play essential role in linking agents while executing common tasks and making their decisions based on environment changes. So, agent belief may be about peer, environment setting, or about another agent. We build DBI ontology so that beliefs can be updated and intentions can be reconsidered during task execution. As we previously mentioned, the framework ontology is developed using OWL-DL [33] ontology language specification. The OWL-DL [33] semantic syntax of any class consists of a set of direct superclasses and a set of restrictions. Then, a class C can be written as **C = {superclasses}** $\cup$ **{restriction}** (here means union). The set of class restrictions include allValuesFrom ($\cup\forall$), someValuesFrom ($\exists$), hasValue (), minCardinality (), maxCardinality (), cardinality (=). While, the class **owl:Thing** is the main class in OWL-DL ontology and every OWL-DL class is a subclass of it.

Agent and DBI classes in our ontology are expressed as the following:

- Agent = {owl:Thing} $\cup$ {*hasBeliefs* $\geqslant$ 1 and *hasDesires* = 1 and *hasRole* some *AgentRole*}. Agent class is expressed as a subclass from owl:Thing and the set of restrictions specify that Agent must own beliefs and desires that the count of each one is more than or equals to 1. Also, the restrictions include specify that Agent class has a role and it is one from the set of roles mention in Fig. 4.
- Belief = {owl:Thing}. Beleif class is expressed as a subclass from owl:Thing and has no defined restrictions.
- PeerBelief = {Belief} $\cup$ {*hasPeer* = 1}. PeerBeleif class is expressed as a subclass from Belief and has a restriction of having only one associated peer.
- AgentBelief = {Belief} $\cup$ {*hasAgent* = 1}. AgentBeleif class is expressed as a subclass from Belief and has a restriction of having only one associated agent.
- Desire = {owl:Thing} $\cup$ {*hasIntentions* $\geqslant$ 1 and *hasPriority* = 1}. Desire class is expressed as a subclass from owl:-Thing and has restrictions that specify having only one associated priority and set of associated intentions that must be more than or equal to 1.
- Intention = {owl:Thing} $\cup$ {*hasMainPlan* = 1 and *hasAlternativePlan* $\geqslant$ 0}. Intention class is expressed as a subclass from owl:Thing and has restrictions that specify having only one associated main plan and another alternative plan that could be empty.
- Plan = {owl:Thing} $\cup$ {*hasType* some *PlanType* and *hasRunningType* some *PlanRunningType* and *hasBodyActions* $\geqslant$ 1 and *hasActionsWhenFail* $\geqslant$ 0 and *hasActions WhenSuccess* $\geqslant$ 0}. Plan class is expressed as a subclass from owl:Thing and has restrictions that specify having plan type(main, or alternative), plan running type (service plan that runs all the time, or action plan that is executed based on action's request), set of body actions to be executed (must be more than or equal to one action), set of actions to be executed on failure (may be empty), and set of actions to be executed on success (may be empty).
- Action = {owl:Thing} $\cup$ {hasWeight some *ActionWeight* and hasState some *ActionState*}. Action class is expressed as a subclass from owl:Thing and has restrictions that specify having action weight (either trivial or vital action), and action state(either unexecuted, done, failed). If any action has weight of vital action and fails to be executed causes the failure of the whole plan and then trying of executing the alternative plans.

Implementing BDI agent model specifies a set of intentions to each agent; these intentions represent the goals that agents try to achieve. Agent intentions are interpreted to set of plans and each plan consists of set of actions need to be executed to fulfill plan and hence agent goals. Actions successful execution is not always guaranteed, as it may require a resource and this resource can not be found because of a failed node or resource access is denied. To recover this, we define alternative plan to be executed if the main plan failed, furthermore plan structure doesn't define only set of actions to be done, but also there are a set of actions associated with it to be executed in case of failure, these actions may contain roll-back of some of the previous actions.

Agents should have the ability to watch the environment in order to respond to dynamic environment changes. Moreover, agent should reconsider their set of desires/intentions and goal relationships with environment updates. In PSG architecture, agents are capable of monitoring environments for specific series of events. These events either are related to application execution logic or environment prosperities. We interpret these events into rules and add them to agent beliefs. Based on these rules, agents can add or infer new beliefs and change their set of desires/intentions according to new beliefs. When a new belief/desire is generated, existing contradictory or obsolete belief/desire will be removed. The following are the rules that agents use in managing their DBI:

When b_{new}, a new belief is formed:
beliefs = *beliefs* $\cup$ $\{b_{new}\}$, and
If $\exists b \in$ *beliefs and* [makeObsolete(b_{new}, b)] then *beliefs* = *beliefs* $-$ $\{b\}$
When a new desire, d_{new}, is formed:
desires = *desires* $\cup$ $\{d_{new}\}$, and
If $d \in$ *desires* and [makeObsolete (d, d_{new})] then *desires* = *desires* $-$ {d}
When a desire is being executed:
actions = getActionsOf(*mainPlan*)
If $\exists a \in$ *actions*[hasWeight(vital) and hasState(failed)] then *actions* = getActionsWhenFail(mainPlan) $\cup$ getActionsOf(*alternativePlan*)

4.3. Framework semantic services

4.3.1. Resource management service

4.3.1.1. Resource indexing. In PSG implementation, we use the Semantic Driven Hashing (SDH) ontology-based indexing scheme for DHT overlay architecture [35]. The basic idea behind SDH is to use the unique identifier assigned to ontology concepts as a key to locate the overlay node responsible for maintaining the resource index associated with the underlying ontology. So, SDH utilizes ontologies, instead of resource names, as the hash input to generate the key necessary to distribute the resource among overlay nodes. When a query to locate or advertise a resource is issued, the SDH scheme obtains the ID of the concepts associated with the resource specified in the query. This ID is then hashed to obtain the key, in the DHT space, of the node where the resource is maintained. This key is then used to route the request to the identified overlay node. The following lines are the pseudo code of the algorithm that is used to locate or advertise a resource:

SDH (Keyword K)
*//*retrieve ontology IDs from the keyword using OMM*//*
OMM_id[ ] $\leftarrow$ *Ontology_discovery(K);*
OMM_keys[ ][ ];
for *i = 0 to OMM _id.length* {
*//*obtain DHT key for each OMM ID*//*
OMM_keys[i][j] $\leftarrow$ *Hash(OMM[i] + j);*}
return *OMM_keys*;

4.3.1.2. Resource accessing. Sharing relationships can vary dynamically over time, in terms of the resources involved, the nature of the access permitted, and the participants to whom access is permitted. These relationships do not necessarily involve an explicitly named set of individuals, but rather they are defined by the access policy that governs access to resources. In order to allow users access resources only those are permitted to them, we attach with each user and resource authority level. Once user announces a resource to be shared, RM agent allows access to be granted to any user when his authority level is greater than resource authority level.

4.3.2. Profiling service

The success of personalized resource discovery depends on its ability to allow users to discover, extract and integrate information of interest from heterogeneous sources, and its ability to provide these users with efficient tools to manipulate and convert the discovered information into knowledge. To achieve this target we build a user profile that concerns with building a closer relationship and understanding of the needs of individuals. User profile stores user interests that are used in resources matchmaking process in a way to provide custom tailoring information to users. Both of resources and user interests are mapped to domain ontology concepts by ontology agent using mapping function. To support semantic-based resource indexing, discovery and advertising ontology agent uses the following model to construct user profile [36]:

- Let **R** be the set of resources in Grid and **K** be the set of domain ontology keywords. Each resource $r \in \mathbf{R}$ is represented semantically by a set of keywords

$$K_r = \{k_i^r \in \mathbf{K}, \quad 1 \leqslant i \leqslant \|R\|\}.$$

- Let **C** represent the set of concept in the domain ontology, S^c is defined as the set of keywords relative to a concept $c \in \mathbf{C}$,

$$S^c = \{k_i^c \in \mathbf{K}, \quad 1 \leqslant i \leqslant \|C\|\}$$

- Resource mapping into concepts: for a given resource $r \in \mathbf{R}$, and for each keyword $k_i^r \in K_r$, to determine the set of concepts C_r associated with resource r,

$$C_r = \{\cup_{iC}^r, \quad 1 \leqslant i \leqslant \|K\|\}$$

- User Profile development: Let R_u be the set of resources owned by user u and C_u be the concepts of interests of user u. Using R_u, a profile P_u obtained as a superset K_u of all keyword sets associated with each resource $r \in R_u$. P_u can be defined as follows:

$$P_u \Longleftrightarrow K_u = \{K_{ri,r} \in R_u, \quad 1 \leqslant i \leqslant \|Ru\|\},$$

and C_u can be defined as follows:

$$C_u = \{C \cup_i^u, \quad 1 \leqslant i \leqslant \|K\|\}$$

4.3.3. Ontology matchmaking service

In our proposed framework, semantic matchmaking is based on domain ontology. When RM agent initiates a resource advertisement/request, a mapping function is used to map the advertisement/request into a domain ontology concept. The degree of match between advertisements and requests is determined by calculating the semantic distance between their concepts. Classes in the OWL ontology are defined by a set of necessary and sufficient condition. In fact, the condition is the semantic description of the class. Hence, the definition distance of two classes is the difference between their semantic descriptions. We use the algorithm proposed by Geo Shu et al. [44] to calculate semantic distance between two concepts. In this algorithm, the semantic distance of two concepts is the sum of the subsumption distance and definition distance.

Any ontology concept can be written as a class **C** that consists of a set of the direct superclasses of C and a set of restrictions,, for short $\mathbf{C} = \mathbf{SS} \cup \mathbf{SR}$. (here $\cup$ means union, while $\cap$ means intersection, **SS** is the short for the set of direct superclasses and **SR** is the set of restrictions). The subsumption distance is the distance between the two concepts in the hierarchy. While, to get the definition distance between two concepts $C1 = SS1 \cup SR1$ and $C2 = SS2 \cup SR2$, the ontology agent performs the following steps:

i. Calculate the Taxonomy Similarity, TS, by the getting intersection between SS1 and SS2.

$$TS = SS1 \cap SS2$$

i. Now, as TS is the common classes in the two concepts definitions, it shouldn't be considered in calculating the definition distance.

$$\text{So, if } TS \neq \varphi \text{ then } SS1 = SS1 - TS \text{ and } SS2 = SS2 - TS$$

i. For every class C in SS1 add its superclasses and restrictions to C1 superclasses and restrictions respectively. So, let $C = SSc \cup SRc$ then $SS1 = (SS1\text{-}C) \cup SSc$, and $SR1 = SR1 \cup SRc$
ii. Repeat the last step for the second concept C2

Now, SR1 and SR2 contain all the restrictions that the two concepts have and inherit from their superclasses. Simply, the definition distance will be the sum of all differences in the SR1 restriction from restrictions in SR2. The meaning of the difference in restrictions here refers to restrictions in SR1 that are not satisfied or realized through the set of restrictions in SR2.

5. Case study

We use PSG framework in constructing semantic campus, which is a semantic P2P application that connects users in order to form a social network of academics in a university. Academic and organizational information that is available by university users is investigated in order to create a campus-based resource described in terms of semantic description. The campus-based resource is also enriched with additional semantics that link one resource to another. Semantic campus application provides a range of capabilities such as ability to diagnose relationships between the academics in the university, ability to find potential experts in specific research areas and ability to provide useful information that represents the individual experience of the academics and research interests that they share.

5.1. Educational campus

A campus network is an autonomous network under the management of a single entity that exists on a university campus or within a local geographic area such as a business park, a government center, a research center, or a medical center. The ideal campus network provides easy access from any access point to all information pools, including library materials, departmental libraries, non-print media collections, institutional databases, etc. The ideal campus network provides easy sharing of electronic resources such as data, text, images, sound, and video across the network. Moreover, the campus network tells a user that his/her friends are nearby. These pervasive applications are active all the time, and move everywhere the user moves.

Universities and colleges are among the most aggressive adopters of Wi-Fi technology. The trend toward more collaborative and open learning environments, fueled by the explosive adoption of mobile devices among students and faculty, makes higher education campuses fertile ground for building wide range of applications that connect academics using wireless LANs. For example, in a university campus students can form small workgroups to keep track of their respective locations, to exchange files and to share presentations and results. Hence, any randomly assembled collection of students' devices such as laptops, personal digital assistants (PDA), or smart mobile phones can be expected to present a highly heterogeneous computing environment.

However, deploying distributed applications on mobile devices and operating using wireless connections impose strong limitations on the design of distributed applications. They need to be able to adapt with the changing conditions quickly, operate in a decentralized way, and consider frequent network connections and disconnections as a rule rather than an exception. Hence, different set of adaptations should be integrated in the distributed applications. These adaptations start with integrating wireless routing routes with application routing tables, integrating stabilization and recovery routines, and finally node ranking mechanisms that classify nodes according to their capabilities and use this ranking in early discovery of weak nodes and thus diminish the costs of sudden node failures.

In order to increase PSG stability and scalability and to be used in developing mobile applications, another profile is developed for peer to store peer information and capabilities including its power, processor, running memory, and churn rate (the rate at which nodes join, and leave system). This peer profile reflects up-to-date peer state and helps in early discovery of week nodes and activating stabilization procedures. Peer

profile is used by agents to rank peer according to its capabilities and to make decisions like allowing this peer to join DHT and working as a *Normal Peer* (stores resources files and keys) or to be *Edge Peer* that only sends services requests and is not responsible for storing any resources. We can write the peer classification as the following:

- Peer = {owl:Thing} ∪ {*hasAgents* ⩾ 1 and *hasAM* = 1 and *hasDF* = 1}
  Where owl:Thing is the root class and every ontology concepts is inherited from it. Every Peer class owns at least two main agents (AM: Agent Management Role and DF: Directory Facilitator)
- NormalPeer = {Peer} ∪ {hasNM = 1 and hasOM = 1 and hasPM = 1 and hasRM ⩾ 1, hasDHT InDHT, ∀hasPower not LowValue, hasConnection not LowValue, ∃∃∃hasComputingResources not LowValue}
  NormalPeer class is expressed as a subclass from Peer and restrictions that specify having agents with roles (NM: Network Management, OM: Ontology Management, PM: Profiling Management, RM: Resource Management). Also, restrictions of NormalPeer specify that this peer is included in DHT and this is the reason it has a network management agent. Moreover, as this peer joins the DHT, its power indicator, connection indicator, and computing resources must not be low.
- EdgePeer = {Peer} ∪ {hasOM = 1 and hasPM = 1, ∀hasDHT OutDHT, ∃hasPower some ValueRestrictions, ∃hasConnection some ValueRestrictions, ∃hasComputing Resources some ValueRestrictions}
  EdgePeer class is expressed as a subclass from Peer and restrictions that specify having agents wit roles (OM: Ontology Management, PM: Profiling Management). As EdgePeer is considered as a week node, it is not included in DHT and hence has no network management or resource management agents. Also, this peer power indicator, connection indicator, and computing resources may have low values.

5.2. Semantic campus

We develop domain ontology using Protégé tool [25] that extends FOAF [34]. FOAF is the abbreviation of Friend-of-a-Friend and is one of the Semantic Web largest and most popular projects. It is essentially Resource Description Format (RDF) vocabulary for describing people and whom they know. We search for available ontologies that describe main entities and activities in university campus. We use the work done by Department of Computer Science at University of Maryland [38,37] and Patrick Gosetti-Murrayjohn [39] as guidelines for building our ontology. Our developed ontology defines elements for describing universities and the activities that occur at them. It includes concepts such as departments, faculty, students, courses, research projects, and publications. As our goal is to analyze internal connections among academics and provide information related to their interest, we use FOAF metadata for describing associations between people and organizations, such as *foaf:knows, foaf:currentProject, foaf:fundedBy*.

In order to provide semantic-based search scheme, concepts that are related to course topics and research interests should be mapped into ontology concepts. Simple Knowledge Organization System (SKOS), introduced by the W3C, is a model for expressing knowledge organization systems in a machine-understandable way, within the framework of the Semantic Web [40]. The SKOS core vocabulary is a set of RDF properties and classes that can be used to express the content and structure of a concept scheme as an RDF graph. The following are sample from semantic campus classes and their properties as developed in our ontology. *sc* is used as a namespace identifier for the semantic campus ontology, FOAF terms are prefixed by *foaf*, and SKOS terms are prefixed by *skos*.

• foaf:Organization	• foaf:Person
– sc:University	– sc:Employee
– sc:Institute	– sc:Student
• sc:Term	– sc:Staff
• skos:Concept	• sc:CampusPlace
• foaf:Document	• sc:Course
• foaf:Group	• sc:CourseInstanceType
– sc:CourseInstance	– sc:Lecture
– sc:ResearchGroup	– sc:Lab
– sc:Project	– sc:Discussion

Sample of Semantic relations: (d and r are used as abbreviations for domain and range)

foaf:member(d = foaf:Group, r = foaf:Person)
foaf:interest(d = foaf:Person, r = foaf:Document)
foaf:publications(d = foaf:Person, r = foaf:Document)
foaf:knows(d = foaf:Person, r = foaf:Person)
skos:semanticRelation (d = skos:Concept, r = skos:Concept)
skos:broader subproperity of skos:semanticRelation
skos:narrower subproperity of skos:semanticRelation
skos:related subproperity of skos:semanticRelation
skos:subject(d = foaf:Document, r = skos:Concept)
advisor(d = sc:Student, r = sc:FacultyStaff)
enrolledIn (d = sc:Student, r = sc:CourseInstance)
interestTopic(d = foaf:Person, r = skos:Concept)
includeResearchTopic(d = sc:Project, r = skos:Concept)
researchGroupInterest(d = sc:ResearchGroup, r = skos:Concept)

Inference rules:

1. knows(student, staff) -> advisor(student, staff)
2. knows($person_i$, $person_j$) -> if member(group,$student_i$) and member(group,$student_j$)
3. knows($student_i$, $student_j$) -> if enrolledIn($student_i$, courseInstance) and enrolledIn($student_j$, courseInstance)
4. interestTopic(person, subject) -> if enrolledIn(person,course) and studiesTopic(course, subject)
5. interestTopic(person, subject) -> if reatcherOf(person,course) and studiesTopic(course, subject)
6. skos:subject(document, $subj_y$) -> if skos:subject(document, $subj_x$) and kos:broader($subj_x$, $subj_y$)
7. studiesTopic(course, $subj_y$) -> if studiesTopic (course, $subj_x$) and skos:broader($subj_x$, $subj_y$)
8. interestTopic(person, $subj_y$) -> if interestTopic (person, $subj_x$) and skos:broader($subj_x$, $subj_y$)

9. researchTopic(group, $subj_y$) -> if researchTopic (group, $subj_x$) and skos:broader($subj_x$, $subj_y$)

6. Evaluation

We build a set of experiments to compute the cost of using ontology in resource matchmaking and indexing. Moreover, to compute the communication overhead we focus our analysis in counting messages exchanged among peers and categorizing these messages based on type or purpose. Besides, as semantic campus application is deployed on mobile devices, we provide recovery and optimized DHT adaptation services. We run a separate experiment to characterize PSG behaviour with different churn rate of mobile peers. The experiments are built as fully symmetric environment which means that all nodes have identical functionalities and responsibilities. But also we use asymmetric capabilities of peers (i.e., different connection states, different battery life values, and different processing capacities of mobile nodes).

So, every peer keeps a profile based on the peer classification in PSG framework ontology. The peer profile is the representation of the peer capabilities at any given time. In the experiments a peer is classified either as edge, or normal node based on its power, connection state, and computing capabilities. In every experiment the peers run concurrently and the peer processing starts by loading the framework services and then join the Chord ring using its ChordID. Moreover, the peers construct the ontology concepts hierarchy from the ontology file and then start to create semantic relation among these concepts and add the relations as ontology beliefs to the ontology agent. When ontology agent receives a new belief, it checks the new belief with the inference rules in order to infer new beliefs if there is a match between added belief and anyone of the inference rules.

6.1. Service time

In this experiment, we want to measure the time taken by each agent role in the framework. The objective of this experiment is to know which service consumes a lot of time and how to minimize this time cost as we know that mobile devices have limited power resources.

6.1.1. Ontology agent

After the peer loads ontology file, ontology agent starts to create the concept hierarchy. In this experiment we measure the time ontology agent takes to build concept hierarchy with different number of concepts at start up of semantic service. Fig. 5 shows the time taken to build ontology tree with different number of concepts and semantic relations.

Moreover, we count number of initial ontology beliefs and inferred beliefs with different number of concepts and semantic relations. Fig. 6 shows the change we got in the number of initial and inferred beliefs with increasing number of concepts and semantic relations.

From the figures above, we can see that both number of concepts and number of relations affects time taken by ontology agent to load ontology and applying inference rules. However, as the number of relations increase, the more beliefs can be inferred from the set of initial beliefs and this match with the fact

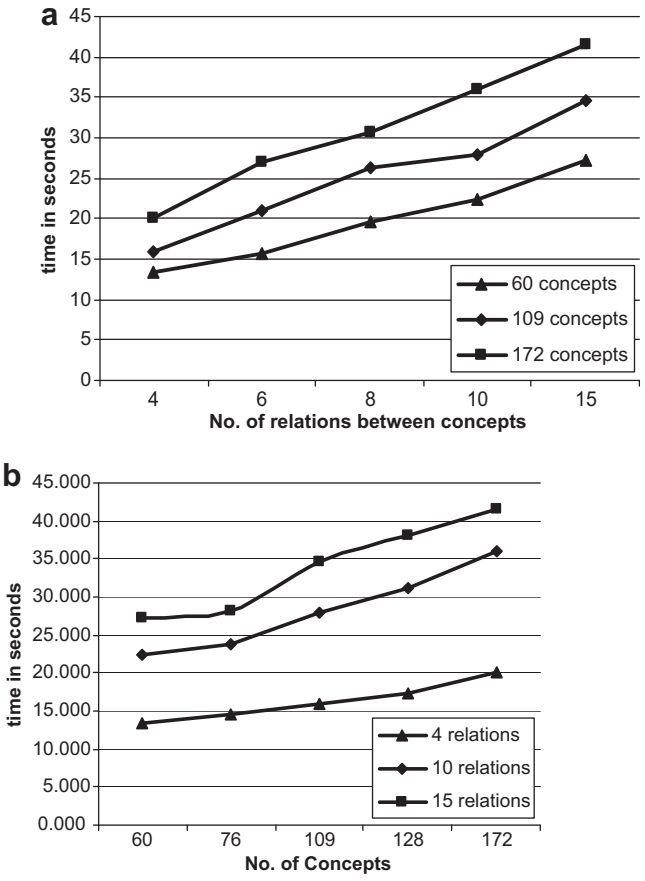

Figure 5 Time with respect to different number of (a) relations and (b) concepts.

that the power of ontologies comes from the ability to describe relations among concepts. Hence, the more semantic relations between ontology concepts, the more reasoning can be done and intelligent results we can get. In order to overcome the time cost of loading ontology, ontology engineers could build set of ontology files with different level of details from the domain ontology and on peer start-up ontology agent can load the appropriate details file that matches the current peer profile.

6.1.2. Resource agent

With respect to resource agent, we measure the time taken to work with different number of resources owned by peers. We measure time taken in publishing resources, and sending resource requests until receiving the resource files. Fig. 7 shows the results of resource agent with different number of resources and in different network sizes (4, 8, and 16 nodes).

The results show that the change in resources count increases the time with lower percentage than the change in network size. As doubling network size from 4 to 8 nodes increases the publishing time with 61:69%, while doubling number of resources from 5 to 10 increases the publish time with 6:13%. Therefore, users of the campus application should take care of the number of resources published on large networks as it would be a time cost process. So, if we attach importance factor with each resource and relate this factor with number of users that will benefit from sharing this resource; this will lead to limit

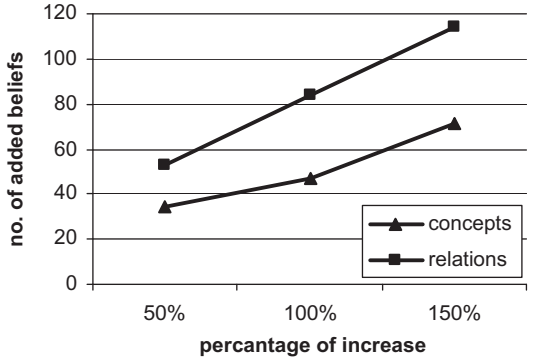

Figure 6 Effect of number of concepts and relations on inferred beliefs.

the number of published resources in large networks. For example, users can give the videos and presentations of the lectures an important factor greater than important factor of the campus social activities videos. Hence, the priority will be assigned to publish videos and presentations of the lectures. Also, we can see from Fig. 7 that the recovery time is almost not affected by number of resources nor the network size. So, we can conclude that the recovery process is not costly and this really matters in mobile domain.

6.2. Ontology match making (OMM)

In Fig. 8, we explore several get requests sent by application in different network sizes (4, 8, and 16 nodes). When user wants to send a get request based on specific ontology concept, the resource agent contacts ontology agent to get the hash value of not only the needed concepts but also its associated and related concepts. Hence, the ontology agent sends the hash values of the requested concept and hash values and semantic distance of the related concepts.

6.3. Communication overhead

The aim of this experiment is to measure the cost of building a structured P2P overlay in mobile ad-hoc networks. We run three different sets of applications, the first uses PSG middleware that offers a structured P2P overlay using Chord DHT over JXME. The second set uses flooding technique over JXME, while the third uses flooding with Time-To-Live (TTL) added to message. TTL is set to the value of chord ring size when the message is created and it is decreased by 1 when the message is received by intermediate nodes. The message is discarded when its TTL reaches 0. We focus our analysis on number of messages exchanged in the following cases:

i. Constructing network overlay.
ii. Advertising a resource.
iii. Requesting a resource.

Before showing the results, we want to illustrate how flooding applications construct their network overlay. We design the first one to work as a mesh network, where each node knows all the other nodes in the network and can communicate directly with them. When node starts, it broadcasts a message to announce its existence to the subsequent nodes. Thus, any node knows all the previously running nodes. However, to construct the mesh the old nodes need to know the new connected nodes. So, when the node is up and connected to the network, it discovers the announcing messages of previously connected nodes, it forwards a message to them, to let them know that there is a new joined node.

In the second flooding application, flooding with TTL, we perform the same scenario but this time we use TTL counter. In this application, the new node sends only one message to its predecessor. The message is sent with TTL counter, and on receiving a message with TTL that is grater than 0, it will be forwarded to the predecessor of the received node. The message will be discarded when its TTL reaches 0.

Fig. 9 shows the communication overhead with the three applications in cases of network set-up, publishing new resource, and sending get requests.

In the above figure, we can see that the Chord overlay has the largest network set-up cost. Also, we notice that the mesh overlay has the highest cost of announcing new resource and it is always equals to number of nodes decreased by 1, as each node knows all the nodes in the network and informs them with the new resource. While this cost differs in TTL and

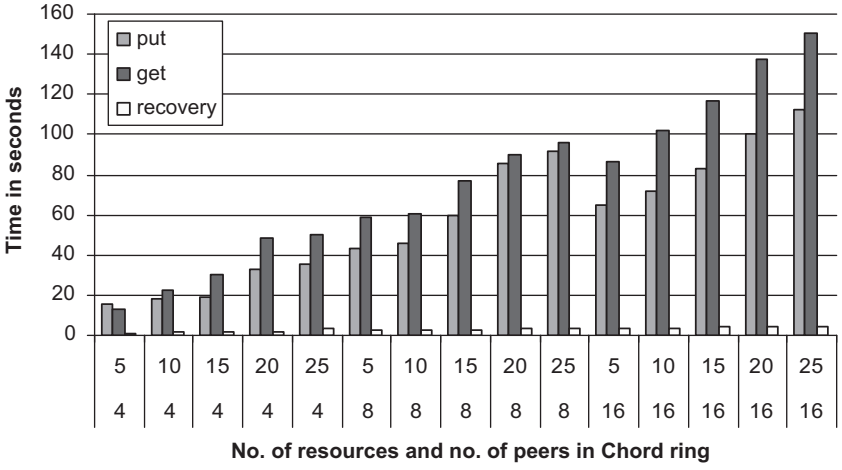

Figure 7 Time taken with resource agent with different network sizes.

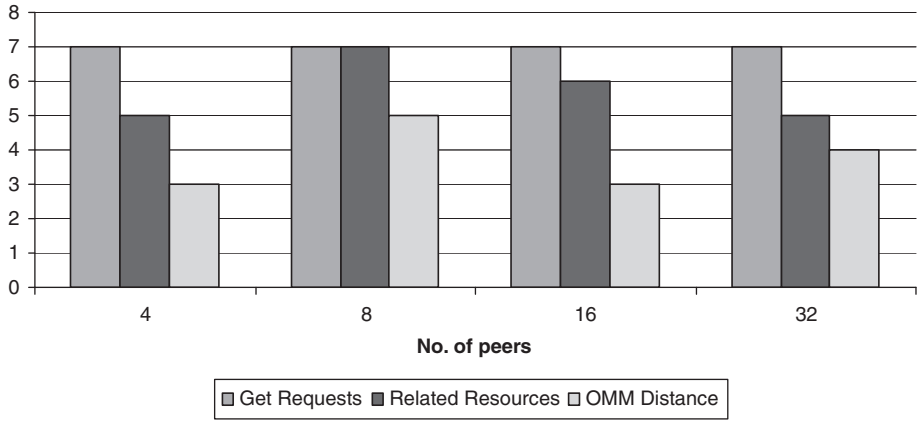

Figure 8 Get requests forwarded and related resources with average semantic distance.

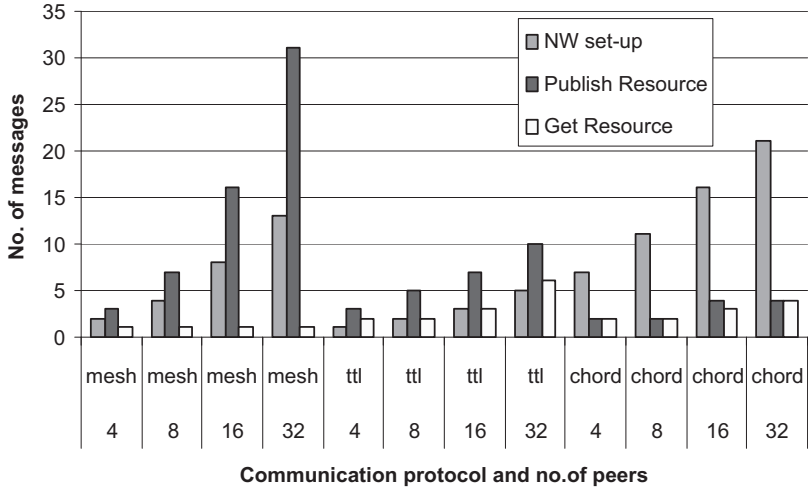

Figure 9 Communication cost with different protocols.

Chord overlays based on the network size. However, the mesh overlay has the lowest get overhead and it always equals to 1 as each node stores the advertisement of resources owned by all the nodes, so when requesting a resource it knows where it is stored. Therefore, we can conclude that chord overlay has the highest network set-up cost but has lower publish and get cost. Moreover, the network set-up cost occurs only once in the life-time of the application, while publishing and requesting resources are repeated processes.

6.4. Stabilization

The aim of this experiment is to characterize the behaviour of PSG with peers churn rate. We run the application with predefined peer ranking and with different percentage of weak nodes. We focus our analysis in:

i. Measuring recovery cost in terms of number of message exchanged to backup weak node resources and DHT entries at its successor.
ii. Measuring cost of chord finger table and successor stabilization in terms of number of message exchanged.
iii. Measuring lost DHT advertisements and resources ratio in cases of failed backups.

In our implementation, when a node gets weak rank, it activates the backup procedure by backing up its DHT entries and resources files at its next live successor. Then it sends a broadcast message; we call it offline_notify; to inform the other nodes that it is going to be offline and the data has been moved to its successor indicating its successorID. For the live nodes in order to fix their finger tables, they send ask_for_predecessor message to successor of any offline node in finger table.

Fig. 10 shows number of recovered DHT advertisements and resources with different rate of weak nodes and different network sizes (4, 8, and 16 nodes). While, Fig. 11 explores the recovery longest back-up paths and number of ask_for_predecessor message sent by live nodes to fix their finger tables. Moreover, Fig. 12 shows the lost DHT advertisements and resources ratio in cases of failed backups with different percentage of weak nodes. In some cases, the lost ratio represents not only the data lost by the current failed node, but also it may contain previously restored data that was owned

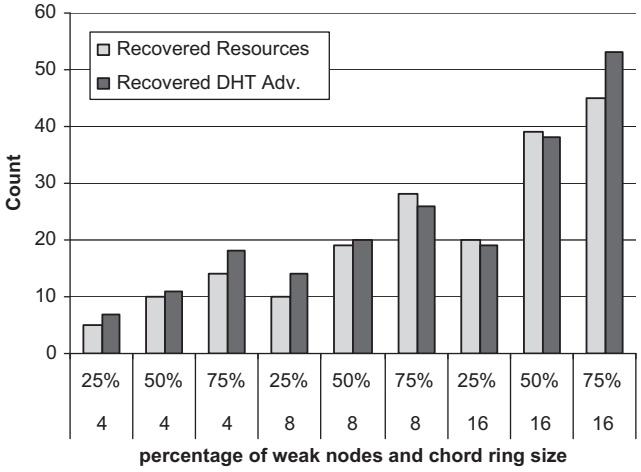

Figure 10 No. of recovered DHT adv and resources with different rate of weak nodes and different network sizes.

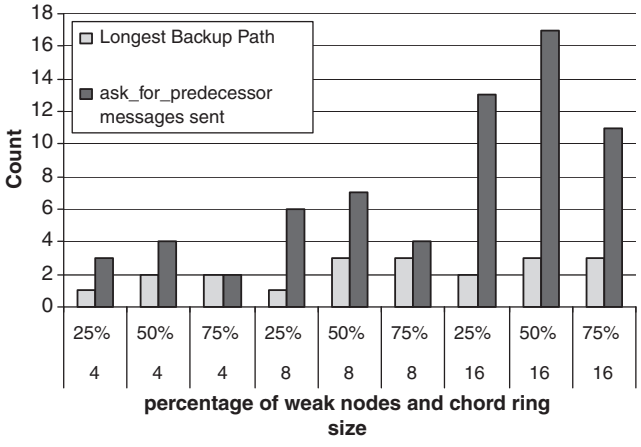

Figure 11 Longest back-up path and no. of messages sent to stabilize finger table with different rate of weak nodes and different network sizes.

by other weak nodes. For the back-up process to fail, one of the following conditions must occur:

1. All the node successors have been offline.
2. The node and its successor are going to be offline at the same time; so only one of them will succeed to back its data.

Fig. 11 explores the recovery longest back-up paths and number of ask_for_predecessor message sent by live nodes to fix their finger tables. The results show that the backup path does not exceed the network size as node's finger table has entries equal to Chord ring size. Also, we can conclude that fixing finger table using ask_for_predecessor message is better than invoking Chord stabilization process on weak node discovery. As the figure shows that total number of ask_for_predecessor messages sent in network size 16% and 50% weak node is 17 messages, while invoking stabilization will cost 128 messages that equal to 8 * 16, where 8 is number of weak nodes and 16 is value of $O(\log N)^2$ that is the stabilization cost.

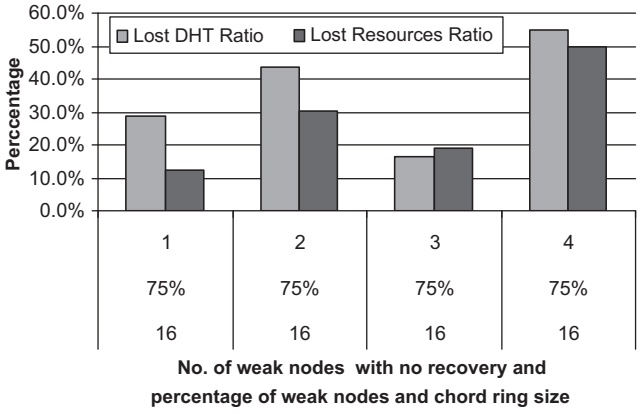

Figure 12 Lost DHT adv. and resources ratios with different weak nodes on network size 16.

Fig. 12 shows the lost DHT advertisements and resources ratio in cases of failed backups (number of nodes where recovery fails: 1, 2, 3 and 4) with 75% of nodes are pre-selected to be weak nodes and the network size is 16.

7. Conclusion and future work

Our work aims to bridge the gap between the conventional Grid computing and its potential application in semantic environments by proposing an agent-based P2P semantic grid architecture. PSG offers a lightweight framework that supports building semantic grid services that are dynamically composed in ad-hoc way, and enables users to form spontaneous services networks. With PSG architecture, users can deploy varieties of semantic applications without support of pre-existing infrastructures or core servers. Users will use only the developed framework prototype integrated with the application ontology. The integration of agents and semantic ontologies results in distributed intelligent system that is significantly more capable, autonomous, and adaptive. Agent roles and hierarchy help to provide modular architecture satisfying requirements like dynamism, self-organizing, reliability, and efficiency. Ontologies are used to help automated processes to access information and enabling both the user and the system to communicate with each other using a common understanding of ontology domain.

PSG framework ontology provides peer ranking service that assists in early detection of weak nodes and moving their data to avoid losing data of disconnected nodes. We admit that PSG has high network set-up cost in large networks, but during network set-up time the node is not idle. Agents run in parallel, so during set-up ontology agent constructs ontology hierarchy and profile agent creates user and peer profiles. Besides, Chord has lower cost in publishing and requesting resources compared with semi-structured and unstructured systems.

For future work, an extension to semantic layer can be made to develop semantic browser that enables users to browse and navigate through available resources and ontology concepts. Furthermore, in case of using more than one domain ontology, the ontology agent must be able to perform ontology alignment process to find relationships between entities belonging to different ontologies. Finally, we want to investi-

gate the applicability of adding task ontology in PSG management layer, and use the framework for parallel task processing.

References

[1] Foster I, Kesselman C, Tuecke S. The anatomy of the grid: enabling scalable virtual organizations. Int J Supercomput Appl 2001;15(3):200–22.

[2] Li G, Sun H, Gao H, Yu H, Cai Y. A survey on wireless grids and clouds. In: Proceedings of the eighth international conference on grid and cooperative computing, Lanzhou, China, 2009. p. 261–7.

[3] Diane C. Evolving data mining into solutions for insights (special issue). Commun ACM 2002;45(8):1–5.

[4] Cannataro M, Talia D. Towards the next-generation grid: a pervasive environment for knowledge-based computing. In: Proceedings of the international conference on information technology: coding and computing, Las Vegas, Nevada, United States, 2003. p. 437–41.

[5] UK EPSRC/DTI Core e-Science Program, <http://www.escience-grid.org.uk/>.

[6] Gruber T. A translation approach to portable ontology specifications. Know Acquisit 1993;5(2):199–220.

[7] Towards Open Grid Services Architecture, <http://www.globus.org/ogsa/>.

[8] Open Grid Forum, <http://www.gridforum.org/>.

[9] Corcho O, Alper P, Kotsiopoulos I, Missier P, Bechhofer S, Goble C. An overview of S-OGSA: a reference semantic grid architecture. J Web Semant 2006;4(2):102–15.

[10] OGDA-DAI, <http://www.ogsadai.org.uk/>.

[11] Talia D, Trunfio P. Toward a synergy between P2P and grids. IEEE Internet Comput 2003;7(4):94–6.

[12] Singh MP. Peering at peer-to-peer computing. IEEE Internet Comput 2001;5(1):4–5.

[13] Harjula E, Ylianttila M, Ala-Kurikka J, Riekki J, Sauvola J. Plug-and-play application platform: towards mobile peer-to-peer. In: Proceedings of the third international conference on mobile and ubiquitous multimedia, College Park, Maryland, New York, United States, 2004. p. 63–69.

[14] Eberspächer J, Schollmeier R, Zöls S, Kunzmann G, Für L. Structured P2P networks in mobile and fixed environments. In: Proceedings of the international working conference on performance modeling and evaluation of heterogeneous networks, Ilkley, West Yorkshire, UK, 2004. p. 1–25.

[15] The Gnutella protocol specification, <http://www.content-networking.com/papers/gnutella-protocol-04.pdf>.

[16] Ratnasamy S, Francis P, Handley M, Karp R, Shenker S. A scalable content-addressable network. In: ACM SIGCOMM conference, San Diego, California, United States, 2001. p. 161–72.

[17] Stoica I, Morris R, Karger D, Kaashoek MF, Balakrishnan H. Chord: a scalable peer-to-peer lookup service for internet applications. In: Proceedings of the conference on applications, technologies, architectures, and protocols for computer communications, San Diego, California, United States, 2001. p. 149–60.

[18] Rowstron A, Druschel P. Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems. In: Proceedings of the IFIP/ACM international conference on distributed systems platforms (middleware), Heidelberg, Germany, 2001. p. 329–50.

[19] Foster I, Jennings NR, Kesselman C. Brain meets brawn: why grid and agents need each other. In: Proceedings of the 3rd international joint conference on autonomous agents and multiagent systems, New York, USA, 2004. p. 8–15.

[20] Jennings NR. An agent-based approach for building complex software systems. Commun ACM 2001;44(4):35–41.

[21] Gil Y. On agents and grids: creating the fabric for a new generation of distributed intelligent systems. J Web Semant 2006;4(2):116–23.

[22] Hayes-Roth B. Architectural foundations for real-time performance in intelligent agents. J Real-Time Syst 1990;2(1):99–125.

[23] Rao A, Georgeff M. BDI agents: from theory to practice. In: Proceedings of the first international conference on multiagent systems, San Francisco, USA, 1995. p. 312–9.

[24] Bratman ME. Intentions, plans, and practical reason. Cambridge, MA: Harvard University Press; 1987.

[25] Noy NF, Sintek M, Decker S, Crubezy M, Fergerson RW, Musen MA. Creating semantic web contents with Protégé-2000. IEEE Intelligent Syst 2001;16(2):60–71.

[26] De Roure D, Jennings NR, Shadbolt N. The semantic grid: past, present and future. Proc IEEE 2005;93(1):669–81.

[27] Cannataro M, Talia D. Semantics and knowledge grids: building the next-generation grid. J IEEE Intell Syst 2004;9(1):56–63.

[28] Gruber T. A translation approach to portable ontology specifications. J Know Acquisit 1993;5(2):199–220.

[29] Gruber T. Toward principles for the design of ontologies used for knowledge sharing. Int J Human-Comput Stud 1995;3(6):907–28.

[30] Gomez AP, Benjamins VR. Overview of knowledge sharing and reuse components: ontologies and problem-solving methods. In: Proceedings of ontology and problem-solving methods: lesson learned and future trends workshop, vol. 18. CEUR Publications, Amsterdam, 1999. p. 1–15.

[31] Uschold M, Gruninger M. Ontologies: principles, methods and applications. J Knowled Eng Rev 1996;11(1):93–136.

[32] JXTA Community, <https://jxta.dev.java.net/>.

[33] Gong L. Project JXTA: a technology overview, <http://www.jxta.org/project/www/docs/TechOverview.pdf>.

[34] Brickley D, Miller L. FOAF vocabulary specification 0.9, <http://xmlns.com/foaf/0.1/>.

[35] Sangpachatanaruk C, Znati T. Semantic driven hashing (SDH): an ontology-based search scheme for the semantic aware network (SA Net). In: Proceedings of fourth international conference on peer-to-peer computing, Zurich, Switzerland, 2004. p. 270–271.

[36] Sangpachatanaruk C, Znati T. A P2P overlay architecture for personalized resource discovery, access, and sharing over the Internet. In: IEEE consumer communications and networking conference, Las Vegas, NV, USA, 2005. p. 24–9.

[37] Simple html Ontology Extension project, <http://www.cs.umd.edu/projects/plus/SHOE/>.

[38] SHOE ontology index, <http://www.cs.umd.edu/projects/plus/SHOE/onts/index.html>.

[39] Patrick MJ. University Ontology, <http://www.patrickgmj.net/project/university-ontology>.

[40] SKOS Simple Knowledge Organization System, <http://www.w3.org/2004/02/skos>.

[41] Meteor project; JXTA-Meteor, <https://jxta-meteor.dev.java.net/>.

[42] Cabri G, Ferrari L, Leonardi L. BRAIN: a framework for flexible role-based interactions in multiagent systems. In: Lecture notes in computer science, vol. 2888 (Springer, 2003). p. 145–61.

[43] FIPA Communicative Act Library Specification, <http://www.fipa.org/specs/fipa00037/>.

[44] Shu G, Rana OF, Avis NJ, Dingfang C. Ontology-based semantic matchmaking approach. J Adv Eng Software, Elsevier 2007;38(1):59–67.