AN OVERLAY ARCHITECTURE FOR PERSONALIZED OBJECT ACCESS AND SHARING IN A PEER-TO-PEER ENVIRONMENT

by

Chatree Sangpachatanaruk

M.S. in Telecommunications, University of Pittsburgh, 1999

B.S. Economics., Thammasat University, Thailand, 1995

Submitted to the Graduate Faculty of

the School of Information Science and Telecommunications in

partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2006

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCE AND TELECOMMUNICATIONS

This dissertation was presented

by

Chatree Sangpachatanaruk

It was defended on

November 28, 2006

and approved by

Dr. Taieb Znati, School of Information Science and Telecommunications

Dr. Martin Weiss, School of Information Science and Telecommunications

Dr. Michael Spring, School of Information Science and Telecommunications

Dr. Kirk Pruhs, Department of Computer Science

Dr. Sujata Banerjee, HP Lab

Dissertation Director: Dr. Taieb Znati, School of Information Science and Telecommunications

ABSTRACT

AN OVERLAY ARCHITECTURE FOR PERSONALIZED OBJECT ACCESS AND SHARING IN A PEER-TO-PEER ENVIRONMENT

Chatree Sangpachatanaruk, PhD

University of Pittsburgh, 2006

Due to its exponential growth and decentralized nature, the Internet has evolved into a chaotic repository, making it difficult for users to discover and access resources of interest to them. As a result, users have to deal with the problem of information overload. The Semantic Web's emergence provides Internet users with the ability to associate explicit, self-described semantics with resources. This ability will facilitate in turn the development of ontology-based resource discovery tools to help users retrieve information in an efficient manner. However, it is widely believed that the Semantic Web of the future will be a complex web of smaller ontologies, mostly created by various groups of web users who share a similar interest, referred to as a Community of Interest.

This thesis proposes a solution to the information overload problem using a user driven framework, referred to as a Personalized Web, that allows individual users to organize themselves into Communities of Interests based on ontologies agreed upon by all community members. Within this framework, users can define and augment their personalized views of the Internet by associating specific properties and attributes to resources and defining constraint-functions and rules that govern the interpretation of the semantics associated with the resources. Such views can then be used to capture the user's interests and integrate these views into a user-defined Personalized Web. As a proof of concept, a Personalized Web

architecture that employs ontology-based semantics and a structured Peer-to-Peer overlay network to provide a foundation of semantically-based resource indexing and advertising is developed.

In order to investigate mechanisms that support the resource advertising and retrieval of the Personalized Web architecture, three agent-driven advertising and retrieval schemes, the Aggressive scheme, the Crawler-based scheme, and the Minimum-Cover-Rule scheme, were implemented and evaluated in both stable and churn environments. In addition to the development of a Personalized Web architecture that deals with typical web resources, this thesis used a case study to explore the potential of the Personalized Web architecture to support future web service workflow applications. The results of this investigation demonstrated that the architecture can support the automation of service discovery, negotiation, and invocation, allowing service consumers to actualize a personalized web service workflow. Further investigation will be required to improve the performance of the automation and allow it to be performed in a secure and robust manner. In order to support the next generation Internet, further exploration will be needed for the development of a Personalized Web that includes ubiquitous and pervasive resources.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

I would like to gratefully and sincerely thank Dr. Taieb Znati, my advisor, for his guidance, understanding, patience, and most importantly his friendship during my graduate studies at the University of Pittsburgh. He became more than an advisor to me. His mentorship was crucial to providing a well-rounded experience consistent with my long-term career goals. He encouraged me to not only grow as a researcher, but also as an independent thinker. I would also like to thank all my colleagues in my research group, especially Anandha Gopalan, Hui Ling, and Guafeng Li for their valuable comments and discussions and for being such good friends all these years.

I would also like to thank my committee including Dr. Michael Spring, Dr. Martin Weiss, and Dr. Sujata Banerjee for such wonderful guidance and suggestions for my thesis. I would like to thank Department of Telecommunications and the Department of Computer Science at the University of Pittsburgh and all of the nice people there for supporting me. I feel at home working there. I would also like to thank all the Ph.D. students in the Department of Telecommunications, especially the Thai students there, for helping me get through difficult times all these years I have spent in the US. I would like to give a special thanks for my special friend, Kirstin Roehrich who is also my dissertation editor. Besides being a good friend, she has taught me so much about how to write clearly and professionally.

I would like to thank my father, Kampol, who is now in heaven, for giving me support and inspiring me to pursue my Ph.D. I would also like to thank my mother, Pannee, for always giving me love and encouragement. Without her love, I could not have come this far. Also, I would like to thank my brothers and sisters back in Thailand, Montree, Maitree, Poltree, Mitree, Kwantree, and Wimolwan for their great support all these years. Finally, and most

importantly, I would like to thank my wife, Ratchanee. Her support, encouragement, quiet patience and unwavering love in the past nine years kept me going and are the key to my success today.

1.0 INTRODUCTION

1.1 PROBLEM AND MOTIVATION

1.1.1 Information Overload Problem

The Internet provides access to a bewilderingly large number and variety of resources, including text, audio and video files, raw scientific data, retail products, transcripts of interactive conversations, and web services. However, due to its scale and decentralized nature, the Internet has evolved into a chaotic repository of all types of information, making it difficult for its users to locate resources of interest to them.

Over the past several years, a number of resource discovery tools have been proposed to facilitate access to information on the Internet by automatically classifying and indexing collections of digital data. In theory, these tools can address the fundamental problem of resource indexing and discovery on the Internet. Yet, it has become clear that they often fail to capture and address the real intent or specific needs of a given user. This failure occurs mostly due to the fact that most of the early search engines and tools, such as bookmark, and directory services, can only provide uniform and equal access to resources in the Internet.

More recent tools, such as the Google search engine, use sophisticated algorithms to assign weights to different web pages. These weights, however, are used merely to rank pages. As such, they do not necessarily reflect the users' personal interests. As a result, queries issued by Internet users often produce an overwhelming number of responses, which frequently contain references to irrelevant material while leaving out more relevant ones.

1.1.2 Semantic Web Evolution and the Community of Interest (CoI)

The recent advent of the Semantic Web, which promises to make the web accessible and understandable, not only to humans, but more importantly to machines [11], is creating opportunities for new approaches and tools to discover resources on the Internet. The Semantic Web makes it possible to associate, with each resource, semantics that can be understood and acted upon by software agents. In particular, web search agents, which can usually be programmed to crawl the web pages in search of keywords, can now be augmented to take advantage of semantic data to carry out sophisticated tasks, such as evaluating the similarity between the web pages, and processing data of interest for users.

In the last couple years, a great deal of Semantic Web research has focused on using "**ontologies**" to annotate data and make it machine-understandable, thereby reducing human involvement in the process of data integration and data understanding [59, 23, 13, 101]. Formally, an ontology is defined as an explicit specification of a shared conceptualization, which can refer to the shared understanding of some domains of interests [42]. As such, an ontology can be viewed as a computational model of a group of related concepts. The model is often captured in the form of a semantic network whose nodes are concepts and whose edges represent relationships or associations among these concepts. The semantic relationships give users an abstract view of an information space for their domains of interest.

The ability to incorporate detailed semantics of data will facilitate greater consistency in its use, understanding and application. Annotating languages and tools are being developed to help users describe resources and model shared agreements of some domains of interest. It is widely believed, however, that the Semantic Web will not be primarily comprised of a few large, consistent ontologies, recognized and accepted by the Internet communities at large [33]. Rather, it is envisioned that the Semantic Web will be a complex web of small ontologies, largely created and used by groups of users who share a similar interest, referred to as "***Communities of Interest***" (CoI). In such an environment, companies, universities, or ad hoc interest groups will be able to link their webs to the ontological content of their interests, thereby allowing computer programs to collect and process web contents, and to exchange information relevant to their needs freely and efficiently.

1.1.3 Enabling a CoI Using the Concept of the Personalized Web

The realization on the web of the CoI ontologies' vision depends on two factors. The first factor is the ability of the Internet users to associate detailed semantics with resources and with different ontologies. The second factor is the availability of adequate tools and mechanisms to allow users to discover, access, and share these resources among their CoI members in an efficient manner.

In this thesis, the concept of a ***Personalized Web*** is considered as central to this vision [80]. It provides the basis for a flexible and effective platform for the development of an autonomous personalization system to enable user-defined views of the Internet. The personalization system will allow users to automatically discover, extract and integrate information and knowledge into a personalized web of interest.

A *user profile driven framework* is proposed to allow individual users to organize themselves into CoIs based on ontologies agreed upon by all community members. In this framework, users are able to define and augment their personal views of the Internet by associating specific properties and attributes to objects and defining constraint functions and rules that govern their interpretation of the semantics associated with these objects. Such views can then be used to capture the user's interests and integrate these views into a user defined *Personalized Web*(PW) [80]. Furthermore, through active advertising and discovery of objects, users automatically expand their "***personal web of interest***" to include other users' personalized webs, thereby enabling the "natural" formation of ***communities of similar interests***.

1.1.4 A P2P Overlay Architecture to Support Personalized Web

Enabling CoIs requires the development of efficient data structures, mechanisms and protocols to support information dissemination, object discovery and object access in a user-transparent manner. Furthermore, in order to support the potentially large and dynamic user communities, the protocols and mechanisms must be scalable and robust.

Recently, the Peer-to-Peer(P2P) overlay technologies have emerged to leverage resource discovery on the Internet. Typically, these technologies allow peers to deploy large-scale, self-evolving infrastructure in an ad-hoc fashion without the need for centralized management or control [73, 90, 106, 78]. The ease of deployment of P2P overlay infrastructure enables the development of a new class of applications that can effectively and strategically distribute their services over the Internet. In this way, they can provide a scalable framework to

develop efficient mechanisms that distributes, shares, and accesses resources in large scale, highly dynamic environments. These properties make a P2P overlay network viable as a solution to support the design of resource discovery for, and enable an effective management of, distributed resource indexes of CoIs.

This research initiative aims to leverage the benefits of the P2P overlay network technology, first by enabling a Personalized Web architecture for a structured and shared access to Internet resources which reflect the user's interests, and by allowing the user to acquire services efficiently and securely as well as share knowledge with members of CoIs. While significant contributions from various fields of research have been made to the evolution of the Semantic Web and P2P overlay networks, to the best of our knowledge, none has established an integrated framework between these technologies to provide a solution for resource discovery, access, and sharing among CoI members. This lack of framework has been one motivation for this dissertation research to explore such a theme.

The remainder of this chapter is organized in the following manners. Section 1.2 describes the objectives and scope of this thesis. In Section 1.3, the formal definition of the concept of a Personalized Web is discussed. The section proceeds with an illustration of this concept using a community of research scientists focused on network security. It is shown that, through document activation, scientists discover new documents of similar focus and form communities of similar interests. In Section 1.4, the functional requirements and the performance objectives of a Personalized Web are described. Section 1.5 introduces the enabling technologies of the Personalized Web, namely agent technology and peer-to-peer overlay networks. The research questions and the fundamental challenges of this research are then discussed in Section 1.6. Next, the approach and contributions of the thesis are described in Section 1.7. The chapter concludes with a description of the organization of this thesis and definitions of the key terms used in this chapter.

1.2 OBJECTIVES AND SCOPE

The main objective of this thesis is to design and develop a P2P overlay architecture to support functionalities of a Personalized Web. This objective is further divided into the following sub-objectives:

- To examine the fundamental principles of a P2P overlay architecture to support ontology-based resource advertising, discovery, access and sharing,
- To design and develop the data structures, protocols, mechanisms, and architecture to enable the Personalized Web in a scalable and robust manner,
- To develop a proof-of-concept implementation of the architecture,
- To assess the performance of the proposed architecture, its mechanisms, and protocols, and
- To develop a case study to demonstrate the proposed architecture to support next generation workflow applications.

This research will not directly focus on issues related to ontologies and their development, creation and representation of user profiles, as well as the development of algorithms to explicitly or implicitly capture users' interests. However, current technologies and tools related to these topics will be used in a case study to demonstrate the concept of the Personalized Web, and assess the performance of the proposed architecture, protocols, and mechanisms.

1.3 PERSONALIZED WEB CONCEPT

A Personalized Web is defined as "*a web of resources augmented by user-defined ontological properties, attributes, constraints and rules, to (i) capture the user's profile and interests, (ii) facilitate user interaction, resource discovery, and access to Internet resources, and (iii) enable creation of communities of similar interests and knowledge sharing among their members*".

To further illustrate the concept of a Personalized Web, consider the case of a research scientist focused on network security, with a specific interest in denial of service (DoS) attacks. Furthermore, we assume the existence of a community of researchers in DoS attacks, whose members include, the *ISAAC Group* at Berkeley, the *Security Group* at CMU, the *NETSEC Group* at the University of Pittsburgh and the *CIS Group* at MIT, as shown in Figure 1.

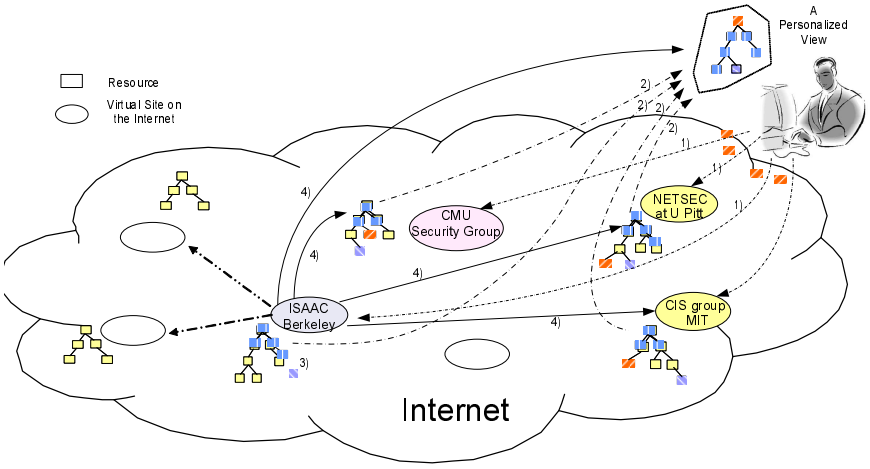

Figure 1: Personalized Webs Enable Communities of Similar Interests

In order to keep abreast of latest advances in the field of DoS attacks, the researcher must be aware of recent publications in the field. Furthermore, the researcher may want to become a member of the DoS research community and share his or her recent results and findings with the members of this research community. The Personalized Web framework, through resource advertising and discovery, enables the researcher to achieve these goals, in an effective and user transparent manner.

The process of discovering relevant publications starts when the researcher associates a similarity profile with a document describing the researcher's specific interests related to the DoS attacks topic. This association can be simply achieved based a set of selected keywords. A more powerful abstract view of the user's interest can be achieved using ontologies to organize keywords and DoS concepts, and capturing semantic relationships among them. Furthermore, the user may define measures which define notions of the user's perception of similarity.

The similarity profile triggers the activation of a semantically-aware agent which proceeds to advertise the user's document and discover similar documents. By using a discovery service, provided by the Personalized Web infrastructure, the agent can locate members of user communities that share similar interests.

Using the user-defined similarity metric, the agent "refines" the Internet related documents into a taxonomy which closely reflects the researchers' interest in the topic. At the same time, the agent advertises the researcher's own publication to the DoS attacks research community. As shown in Figure 1 (step 1), the agent uses a user-defined similarity metric to determine if a discovered document is of interest to the user. At the same time, the agent advertises the researcher's document to the DoS attack community (step 2). Documents, deemed similar based on the user-defined similarity metric, are used to build the user's personalized view of DoS attack documents of interest.

As new documents are discovered, the user's personalized web of DoS attack documents evolves dynamically and adaptively to reflect changes both in the Internet environment and the user's interests. As shown in Figure 1, the addition of a new document to the *ISAAC Group* document tree, (step 3), causes the agent to notify other members of the community about this addition, (step 4). Based on their own similarity metrics, members may either choose to add this document to their own document tree, if the new document is deemed of interest, or ignore the advertisement. In former case, the user's Personalized Web of DoS attack documents is updated accordingly.

1.4 BASIC FUNCTIONALITIES AND PERFORMANCE OBJECTIVES OF THE PERSONALIZED WEB

The previous section conceptually demonstrated how a Personalized Web facilitates user interaction, resource discovery, and access, and enables knowledge sharing among members of CoIs. To enable the vision of a Personalized Web, architecture, protocols and mechanisms must be developed to:

1. Enable resource discovery based on user specific profiles, associated with ontologies of interest, in a user-transparent manner.
    - This functionality allows a user to discover resources that reflect the user's interest in a transparent manner. The user's interest can be expressed explicitly, using a profile language, or implicitly by capturing and analyzing user interaction with the Internet. Resource discovery is achieved based on a user-defined, ontology-based similarity. Semantic similarity among resources may be based on statistical information in the concepts and attributes of the user defined ontology and their relationship within the ontology. Different methods for measuring similarity between resources can then be used. Traditionally, methods for measuring similarity between concepts and attributes can be based on measuring the distance, expressed in terms of the number of edges between the node associated with concepts or attributes. Variations of these methods may associate weights to the links to reflect their positions in the taxonomy. Information-theoretic models which consider relations, such as "is-a", and "has-part", may also be used. These methods have proven to be less sensitive to density variability [75].
2. Enable user-selective resource advertising among user communities of similar interests.
    - This functionality allows users to advertise their resources to users who are in their specific domain of interest. Resource advertising is achieved based on a user-defined advertising profile. The profile includes a user-defined ontology, and an advertising scope and policy. Ontology-based similarity is used to find users who are interested in the resources associated with the ontology, similar to the discovery. The scope is used to limit the advertising to specific domains of interest, such as an Internet domain, a user community or an interest group. The advertising policy defines the advertising strategy, which can be either *aggressive*, whereby the service proceeds to advertise its associated resource immediately, or *passive*, whereby the service waits for other users to discovered the resource.
3. Enable ad hoc interest groups to form and dynamically expand, by recruiting new members with similar interests.
    - This functionality allows CoIs to develop dynamically through resource discovery and

advertising. On the one hand, the resource discovery allows users to find CoIs where they should retrieve resources from, since they are interested in these CoIs members' resources. On the other hand, the resource advertising allows users to discover CoIs where they could share resources, since these CoIs members are interested in their resources. CoIs are developing independently and evolving through time as members join and leave. The similarity of the ontology associated with user resources is used to self-administer members of CoIs.

User-based resource discovery coupled with resource advertising allows users to share resources and expand their findings among users who have a similar interest. They facilitate users to form CoIs. Consequently, both user-based discovery and resource advertising enable the knowledge sharing that continues to evolve through time among users who share similar interests.

1.4.1 Performance Objectives

To provide these Personalized Web services effectively over the Internet, performance requirements need to be addressed. The Personalized Web must be capable of accommodating both the user's needs as they evolve, and the community as it grows. Furthermore, the architecture must be able to cope, without a severe performance degradation, with a potentially large number of resources. To address these requirements, the Personalized Web architecture must be ***scalable***.

Scalablability

Typically, the scalability of a system can be evaluated with respect to a set of independent *parameters*, observable *metrics*, and the *environment*. A *parameter* reflects specific aspects of the system and can be freely varied. An observable *metric* is evaluated as the system is scaled with respect to the independent parameters. A set of *environment* parameters depend on the application and reflect the design choices of the system. The scalability of a Personalized Web architecture and its underlying protocols and mechanisms can be expressed in terms of a set of independent parameters, including *the number of overlay nodes, the number of resources, the number of users, the amount of storage in each node, the rate of generated queries* and

the load of each nodes. The main observable metrics of a Personalized Web include the *degree similarity* between the retrieved resources and the user's interest, *the query response time*, *the number of routing steps* and *the communication overhead*. The environment parameters include the *network characteristics, the types of users* and the *types of user queries*. The scalability of the Personalized Web is a measure of the ability of the architecture and its associated protocols to maintain efficiency, while a subset of these parameters varies over a range of potentially large values.

Another important design requirement of the Personalized Web architecture is its ability to function correctly and efficiently in the presence of failures. To meet this requirement, the Personalized Web architecture must be ***robust***.

Robustness

The robustness can be measured with respect to a given set of *system features*, with a set of *perturbations* applied to the system [3]. The system features of the Personalized Web include *data accessibility* [79], and *response time in retrieval and advertising*. The perturbations include *node leaving and entering the network*, *node failure*, and *network disruption*. The robustness of the Personalized Web is a measure of the ability of the architecture and its associated protocols to maintain Personalized Web data, despite fluctuation in the behaviors of the components, such as an overlay node or the underlying network.

1.5 ENABLING TECHNOLOGIES

To address the fundamental requirements of Personalized Web architecture, this work leverages the benefits of the *P2P overlay networking* and agent technologies [7]. The term "*agent*", in the context of this thesis, is defined as a computational entity which:

- Acts on behalf of the user, in an autonomous and transparent fashion,
- Performs its actions within a context defined by the user, either proactivily or reactively, depending on the context, and
- Exhibits a level of learning and cooperation with its peers commensurate with user-defined boundaries.

Agent technology is a paradigm of computation whose applications from the onset have exhibited both its power and diversity. Combined with P2P technology, an agent-driven framework provides the foundation for a large-scale, self-evolving ad-hoc infrastructure, without the need for centralized management or control [87]. The ease of deployment of P2P overlay infrastructure enables the development of a new class of applications that can effectively and strategically provide services over the Internet. This benefit of agent technology has been exploited by several large scale file sharing systems, including Gnutella, Morpheus, and Freenet [40, 64, 64, 37]. P2P overlay networks have also been used to support other types of self-organizing Internet applications, including resource sharing, ad-hoc communication and collaboration, improved resiliency and robustness [85, 41, 7].

P2P technologies can be leveraged to enable a Personalized Web infrastructure and support its basic functionalities. The P2P overlay networking infrastructure provides a scalable framework to develop efficient mechanisms to distribute, share, and access resources in large scale, highly dynamic environments. Furthermore, P2P technologies offers the potential to organize resources into a collection of logical overlay networks; each logical overlay network reflects the interest of a specific community whose members share common interests. Note that the same physical resource can be accessible from different logical overlay networks, but its semantic attributes may vary depending on which overlay network it is being accessed from. In the following, the scope of this thesis is described by discussing the fundamental challenges and their related issues of the design and development of a Personalized Web using the P2P overlay technologies.

1.6 RESEARCH QUESTIONS AND FUNDAMENTAL CHALLENGES OF THE PERSONALIZED WEB

In order to realize the functionalities of a Personalized Web by leveraging P2P overlay technologies, several fundamental questions need to be addressed:

- Is it possible to combine the P2P overlay network and agent-enabled Semantic Web technologies to enable a Personalized Web?

- If so, what mechanisms, protocols, and data structures must be in place to support personalized resource discovery, advertising, and access on the Internet?
- How can these functionalities be provided in a scalable, and robust manner?
- Can these mechanisms, protocols, and data structures be extended to support Web Service Workflow applications?

Answering the above questions is the focus of this research thesis. Before discussing these questions, the fundamental challenges to enabling a Personalized Web are addressed. The focus of the discussion will be on the design and development of the architecture, its mechanisms, protocols, and data structures to enable resource discovery, resource advertising, and dynamic CoIs on the Internet.

1.6.1 Resource Discovery

The first challenge is to facilitate resource discovery based on user-specific interest profiles in a scalable and robust manner. In the example above, the discovery service is assumed to exist in order to inform the agent where to seek for resources that are related to the scientist's interest. In fact, to provide this service efficiently using the P2P overlay network technology, the architecture must establish mechanisms to detail semantics of resources hosted among peers. These mechanisms facilitate agents to find interests in these resources. In addition, the architecture must collaborate in indexing these semantics in order to provide efficient semantically-based resource location and retrieval. As such, agents can effectively discover these resources, and refine them based on the users' interests.

Furthermore, the architecture must be able to support this functionality in an Internet environment where numbers of overlay nodes, resources, and users are growing quickly, and the churn rates[1] are high. The architecture, mechanisms, and protocols must provide a resource discovery service that is both scalable and robust in such an environment.

[1]The rate of entering and leaving the network.

1.6.2 Resource Advertising

The second challenge is to facilitate agents to advertise the user's resources in a scalable and robust manner. The advertising mechanism is used in a Personalized Web to disseminate resource information to users in a community. When a user posts a new resource, an agent is activated to discover the community that may have interest in this resource, and advertise resource information to the members in the community.

Like the resource discovery, the architecture must support resource advertising in a scalable and robust manner. The mechanism and protocol must allow agents to advertise efficiently among communities members, even when the size of the community is very large and the churn rate are very high. Both the mechanisms and the protocols have to be robust when some of the overlay nodes fail or a network disruption occurs.

1.6.3 Communities with Similar Interests

The challenge is to facilitate communities with similar interests to develop efficiently in order to meet the different needs of Internet users. To realize this concept, the architecture must facilitate agents to create, join, leave, and manage the communities with similar interests based on the users' interests. When users have interests in common, the protocols, and mechanisms must allow their agents to discover each other to share their resources and findings.

The abilities of the P2P overlay technologies can be used to support the development of logical overlay to serve different needs of users in a community. However, since each user can independently form a community of interest, the protocols, mechanisms, and data structures must be able to support different needs of members within a community. For example, when user a is interested in resources R_b of user b, but resource R_a of user a does not interest user b; user a wants to join user b's community, yet user b wants to separate from user a's community.

From this simple scenario, one can expect that as the communities grow in size and continue to be self-adaptive in order to fulfill the dynamic needs of users, the mechanisms and data structures will become increasingly complex. They have to not only allow each user

to manage its own logical overlay network, but also to support collaboration among users. When the similarity between communities grows, they must also allow the communities to merge for efficiency in resource sharing. On the other hand, if a conflict develops within a community, they must allow the community to be split in order to effectively serve different needs of users. In addition, when an agent joins or leaves a community, they must allow the community to self-organize to reflect the changes in a user transparent manner.

1.7 APPROACH AND CONTRIBUTIONS

The objective of this thesis is to develop the Personalized Web architecture, its mechanisms, tools, protocols, and data structures to support personalized object discovery, access, and sharing. In order to meet the functional requirements of the Personalized Web, the mechanisms, tools, and protocols must be efficient, scalable and robust.

The core of the proposed methodology to enable the functionalities of a Personalized Web technology and address its performance requirements is based on the abstraction of an Internet object as an active *agent*, which, aimed with *ontological* and *semantic* capabilities and *similarity metrics*, interacts with its *peers* through advertising and discovery. This interaction forms the basis for a user-defined, agent-enabled personalized semantic web, which reflects the user's views of the Internet.

The proposed approach to enable a *Personalized Web* is to leverage the benefits of two technologies, namely the *agent-enabled Semantic Web*, and *P2P overlay networking* in a unified framework for flexible and scalable resource advertising and discovery. The main idea is to use agent-enabled Semantic Web technology to associate semantic structures to objects based on user-defined interests. A P2P overlay network is then used to integrate these semantic structures into an efficient object indexing and location overlay structure for scalable semantically-based object discovery and access. The resulting framework will provide the foundation upon which to design and develop a Personalized Web architecture, mechanisms and protocols. The challenge remains as to what mechanisms and protocols must be in place to accommodate a wide range of user profiles and Internet objects in a scalable and efficient manner.

In summary, the contributions of this thesis are as follows:

1. A Personalized Web Framework that
   - Enables ontology-based resource discovery, access, and sharing among members of CoIs,
   - Enhances resource discovery through user specified rules and similarity metrics to capture the users interest,
   - Utilizes an agent-driven "active object advertising and retrieval" to enable the Personalized Web in a user transparent manner,
   - Achieves scalability and robustness of resource discovery and advertising through an underlying P2P overlay network.
2. A prototype implementation of Personalized Web architecture.
3. A hybrid emulation-simulation framework to assess performance of Personalized Web architecture, protocols, and mechanisms.
4. An extension of the Personalized Web to support Web Service Workflow applications.

1.8 ORGANIZATION

The rest of this thesis is organized in the following manner. Chapter 2 reviews the literature in the field of resource discovery deploying P2P overlay networks. The literature is organized based on the classification of the underlying P2P overlay substrates: unstructured and structured P2P overlays. The objective in this organization is to give the reader an overview of the state of the art in the field and to identify the major shortcomings of existing works regarding the infrastructure in order to enable the Personalized Web.

In Chapter 3, an overview of the Personalized Web architecture will be introduced. Conceptually, the Personalized Web architecture is composed of three main layers. The top layer deals with semantic-aware agents, referred to as "***semlet***." These agents act on behalf of users to advertise and retrieve objects of interest in order to develop Personalized Webs. The middle layer is composed of an overlay network referred to as ***Ontology Overlay Network (OON)***. The OON provides a scalable and robust infrastructure for object storing,

indexing, lookup, and location. It facilitates semlets to advertise and retrieve objects among CoIs. The lower layer deals with the low-level object addressing provided by the Internet infrastructure. The key components of each layer and how they support Personalized Web functionalities are also discussed in this Chapter.

In Chapter 4, object advertising and retrieval schemes supported by Personalized Web architecture will be discussed. The schemes, namely *Aggressive* scheme, *Crawler-based* scheme and *Min-Cover-Rule* scheme, are described. These schemes are designed with different performance objectives, which give different benefits and drawbacks in various circumstances. Such circumstances are also explored in this chapter.

Chapter 5 presents the experimental framework to demonstrate "the proof of concept" implementation and to evaluate the schemes introduced in Chapter 4. The framework includes both emulation and simulation based experiments. An emulation is proven to be a high-level proof-of-concept implementation with its ability to interact with real processes (i.e. semlet and OON node processes) within an emulated network. As such, the emulation is used to demonstrate a proof of implementation and deployment of the Semlet and OON. However, the emulation lacks scalability and flexibility. It does not handle well for a large number of processes or for an environment where the network is changed frequently. The simulation-based experiment is then used to address these problems. It is employed to evaluate the performance of the Personalized Web architecture in a large scale network, and in a churn environment. In this framework, the main matrices used to evaluate the schemes include the effectiveness and efficiency of the schemes. Effectiveness is measured in terms of precision and recall of object advertising and retrieval[2]. On the other hand, efficiency is assessed in terms of communication and storage costs.

Chapter 6 presents a case study which extends the Personalized Web framework from dealing with static resources to dynamic resources provided by "Web services" to support the next generation "workflow" applications. The main objective is to demonstrate the ability of Personalized Web architecture to support diverse Internet-based "workflow" applications that demand for automation of object discovery, access, and sharing among communities of consumers and providers. The chapter describes the concepts of Web services, OON for Web

[2]The terms used here are slightly different from the ones used in Information Retrieval community.

service registry referred to as ***Ontology-based Registry Overlay Network (ORON)***, and the semlet agent for Web service workflow, referred to as ***flowlet***. Additionally, the chapter discusses the mechanisms for Web service advertising, discovery, and orchestration to achieve a Web service workflow.

Finally, Chapter 7 concludes this thesis and discusses possible future research directions. The discussion of research directions focus on two main directions: a way to integrate techniques proposed by related research to improve the mechanisms that develop the Personalized Web and, secondly, a way to extend the Personalized Web architecture to support personalized pervasive and ubiquitous resources and services within CoIs.

1.9 DEFINITIONS

Definition 1. Community of Interest(CoI)*: An ad-hoc group of users who share a similar interest in a particular domain. Members in a CoI are assumed to have a shared ontological foundation of a conceptual model. As such, this model is used to refer to a set of semantics of interest within the CoI.*

Definition 2. Ontology*: An explicit specification of a conceptualization, which defines how things are related within a domain of interest. An ontology structure can be viewed as a semantic network of concepts, where the edges connecting them define their relations. Several types of relations may be defined for an ontology. In this thesis, the hyponymy relation is mostly used. A hyponymy-based ontology relates more specific concepts to more general ones (from which generic information can be inherited). Such links have been variously named "is a," "member of," "subconcept of," "superconcept," etc. Such links are used to organize concepts into a hierarchy or some other partial ordering, called a "taxonomy". More general concepts in such a partial order are said to subsume more specific concepts, and a more specific concept is said to inherit information from the subsumers[14].*

Definition 3. Ontology-based Resource Advertising*: The mechanisms used to inform users who have an interest in the resources using the ontology-based profiles when they are*

available. The advertised information includes resource location and the semantics they are representing.

Definition 4. Ontology-based Resource Discovery*: The mechanisms implemented to discover the resources using ontology-based profiles. Ontology is used to represent semantics of resources and search queries, and to allow the semantic distance among them to be inferred and measured in an effective manner. For example, a "car" would relate to "automobile" and a "car" would also relate to "motorcycle" since they both are kinds of "vehicle".*

Definition 5. P2P Overlay Network*: a virtual network that is strategically formed by a group of peers*[3] *which may or may not be physically connected. Each peer can play dynamic roles: client, server, or router, based on the functionalities required among peers in order to facilitate network functionalities and enable P2P applications and services to perform their intended operations transparently regardless of their physical locations, logical domains, and IP addresses.*

Definition 6. Web Service*: A self-contained, self-describing modular application that can be published, located, and invoked across the Web [93].*

Definition 7. Web Service Workflow*: The automation of a web service process, during which data input and output are passed from one Web Service to another for action, according to the set of rules defined by a given Workflow and Web Services specification.*

[3]The terms "*peer*" and "*node*" are used interchangeably in this work. They both refer to a physical machine connected to the network.

2.0 LITERATURE REVIEW

In recent years, P2P overlay networks have gained popularity, and impacted a lot of users, businesses and research communities. Like the web, P2P networking grew rapidly out of user desire. Whether sharing music, pictures, or conversation, the user desiring to connect on a peer-to-peer basis without the intervention of a centralized governing authority, seems to express a fundamental human need [53].

The P2P overlay networks offer attractive properties, including flexibility, ease of deployment, and scalability. They allow users to join and leave a network easily, while providing mechanism to cope with the change efficiently and systematically. In addition, the maintenance costs for these networks in terms of CPU cycles and network communications are relatively low and manageable, compared to those that implement directly on the IP network. A peer in these overlay networks, is required to maintain very little information about a network in order to be able to locate resources on a large network. Because of these attractive properties, P2P overlay networking is employed as the foundation of the Personalized Web architecture.

The following sections describe the state of the art of the P2P overlay architectures proposed in recent literature as well as discuss their protocols and the mechanisms that support the functionalities of the Personalized Web. The goal of the discussion is to answer the first research challenge: "*Is it possible to combine P2P overlay network and agent-enabled Semantic Web technologies to enable a Personalized Web?*" To answer this question several aspects of the technology must be explored. First, the P2P overlay networks, their definitions and developments are explained. Two typical classes of P2P overlay networks, namely unstructured and structured are described, followed by a discussion of the proposed architectures in terms of resource discovery, access, and sharing. The purpose of the discussion is to examine the possibility of using them to implement the Personalized Web. The chapter concludes with an argument about using the proposed architectures to enable the Personalized Web.

2.1 INTRODUCTION TO PEER-TO-PEER(P2P) OVERLAY NETWORKS

The term "peer-to-peer" was originally used in the mid-1980s by local area network (LAN) vendors to describe their connectivity architecture. In essence, peer-to-peer simply means *equal communicating with equal* [53]. However, the general term of P2P today is defined in a more specific manner with regards to the Internet. It is defined as "a class of applications that takes advantage of resources – storage, cycles, content, human presence – available at the edge of the Internet" [87].

In the context of this thesis, the P2P is viewed as a communications model, in which each peer[1] can play dynamic roles: client, server, or router based on the required functionalities among peers. P2P enables the Internet to be the platform for sharing resources on a larger scale without the constraints of a management structure. The *overlay network* is a virtual network on top of the existing network. Overlay network technologies facilitate network functionalities and enable P2P applications and services to perform their intended operations, transparent to their physical locations and logical network domains. To examine the properties, mechanisms, and evolutions of these applications and services, two main P2P overlay categories namely unstructured and structured P2P overlays are reviewed in the following sections. Each section starts with the background of each approach, followed by discussions of the literature related to resource discovery.

2.2 UNSTRUCTURED P2P OVERLAY NETWORKS

A P2P overlay network formed without any global structure to regulate search and neighbor selection is called an *unstructured overlay network*. It became well-known in a short period of time due to its use as a support for popular file-sharing applications on the Internet. An unstructured P2P overlay network has been employed to provide two main functions. First, it allows users' nodes to conveniently join and leave the network without disruption of

[1]The terms "*peer*" and *node*" are used interchangeably in this work. They both refer to a physical machine connected to the network.

network operations. Second, it enables users to efficiently search and share files located on their hosts. The ease of deployment and scalability of the unstructured P2P overlay network, demonstrated by these file-sharing applications, have inspired networking research to explore the possibilities of supporting future networked applications. One of the main topics to explore was resource discovery. There are three main approaches used to accommodate resource discovery in the unstructured overlay network: the *centralized index* approach, the *pure decentralized* approach, and the *hybrid, super-peer* approach.

The centralized index approach was first introduced by Napster, a centralized-index-based file-sharing system. Napster deployed a group of servers to allow peers to register their local resources to and search for remote files from the Napster network. When peers wanted to share their local files, they sent information about their files to these servers. These servers then indexed the files' information with the current IP addresses of the peers, allowing other peers to search for these information. When peers wanted to search for files, they sent a query that contains file information such as a file name, an author name, or a set of keywords to Napster's servers. These servers then returned the addresses of the peers that had the matching files or otherwise returned "not found". These servers provided a simple, flexible search for the locations of the files based on the file information.

Despite the simplicity in the deployment and the ease of management of the centralized approach, it has a single point of failure and scalability issues. When the centralized servers fail, the rest of the Napster network cannot operate. Additionally, when the numbers of peers, resources, and users become very large, the centralized servers would not be able to store all the resource indices or handle all the queries.

The pure decentralized approach, used by Gnutella and its variants, were proposed as an alternative infrastructure that removed a single point of failure and improved the scalability of the centralized approach. Each peer in a pure-decentralized P2P overlay network maintains an index of its local files. To search for files on the network, a peer starts a blind search by sending a query to all of its neighbors. The nodes that receive the query then continue forwarding information to its non-ingress neighbors[2] until the search criteria, such as number of propagating hops or number of discovered files, is satisfied [40].

[2]The neighbor node that sends the query to the node is the ingress neighbor.

The pure decentralized approach eliminates the single point of failure and improves scalability of the search, yet its blind search mechanism can be costly in terms of network bandwidth. For example, a network where each node has four neighbors on average and there is a ten hop limit for the search criteria can generate up to 19,687 queries[3] for a query. Because of the criteria of the blind search, the pure decentralized approach operates within a limited scope[4]. It is possible that existing resources could be left undiscovered if queries do not reach the hosts that have the matching resources. The success of a search depends on both the distribution of resources among the overlay peers and the location of the starting peer.

The hybrid approach, namely the super-peer-based approach, leverages benefits from both the centralized and the pure decentralized approaches. It employs a subset of overlay nodes referred to as *search hubs* to perform resource indexing for resources shared by a group of peers. These search hubs connect to one another using the pure decentralized approach which allows them to support the search for resources indexed both locally and remotely. A search starts by sending a query to the local hub. If a matching resource is not found, the search continues using a blind search proceeding from the local hub to other hubs. The search continues until search criteria, similar to those of the pure-decentralized approach, are satisfied [52, 64].

The super-peer-based approach achieves better scalability at a higher cost of supper-peer management. This approach implements mechanisms that select super-peers in the network, allow peers to discover and connect to a super-peer, and handle resource indexing when a super-peer leaves the network. Furthermore, this approach still uses a blind search to locate resources. As a result, it can provide inconsistent search outcomes. Figure 2 illustrates the overview of the search processes of these three approaches.

[3] The upper bound on number of queries for a network with average x neighbors and y hops limit is computed by the formula: $x + (x-1)^{(y-1)}$. The upper bound on number of queries for a network with 4 neighbors on average and a 10-hop limit is computed by $4 + 3^9 = 19,687$.

[4] The scope refers to the ability to locate existing resources in the network.

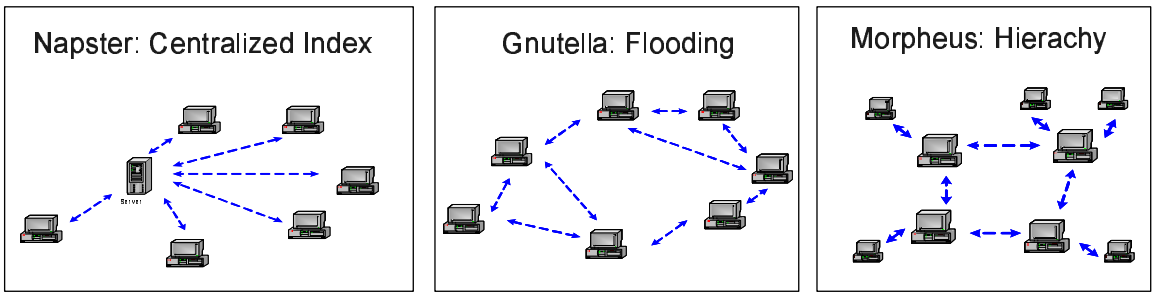

Figure 2: Unstructured P2P Overlay Systems.

2.2.1 Resource Discovery in Unstructured P2P Overlay Networks

This section reviews the recent literature that propose architectures, data structures, mechanisms, or protocols to improve resource discovery in an unstructured P2P network. Based on the popularity of the topics in the recent literature, the review is organized into two main sections: mechanisms to support richer semantics and mechanisms to increase the efficiency of the search based on these mechanisms.

2.2.1.1 Mechanisms To Support Richer Semantics Typically, a resource in a P2P system is represented by a simple group of semantics, such as a set of keywords or metadata in the form of attributes and values. This representation is used to associate resources in the search index. To search for a resource associated with a specific keyword, a searcher has to send a query with this same keyword. Despite its simplicity, the system does not allow users to find resources based on semantics, instead on the word forms that are indexed. As a result, when a searcher sends a query with other similar keywords, the searcher can not find that resource. On the other hand, when a searcher sends a query with a specified keyword, the searcher may find unrelated resources that are indexed with the same word forms, yet in different contexts. These problems have been recognized in the Information Retrieval (IR) community as the *synonymy* and *polysemy* problems [38]. The term synonymy refers to the situation where many terms have the same meaning. Retrieving resources that only contain the same word as in query term would miss those resources that contain different

terms. On the other hand, the term polysymy refers to the situation where one word has many different meanings. In contrast to synonymy, resources may be retrieved even though they are irrelevant. Several techniques that use conceptual relations, including therauri, taxonomies, and ontologies have been proposed to address the problems of synonymy and polysymy.

To alleviate the synonymy and polysymy problems of resource discovery in a P2P environment, conceptual relations in the form of ontologies and taxonomies are used to explicitly classify semantics of resources and queries. Resources and queries are associated with ontologies, so that their meanings can be computationally and automatically inferred from their descriptions. As a result, peers can identify relations between queries and resources more accurately and in a user transparent manner [20, 8, 1, 81].

Using ontologies in a P2P network requires extra mechanisms to maintain their information and to allow users to access and acquire them in an efficient manner. The proposed solutions either employ a fixed, global knowledge that all users know *a priori*, or an adaptive, adhoc concept-mapping through an inference engine. The approach based on a global knowledge simplifies the mechanisms to deal with the inconsistency and change of knowledge. All the meanings and relations of concepts used to describe resources and queries are defined in the global ontology. All users employ this ontology as a central semantic reference which allows them to search and discover resources in a consistent manner. Figure 3 illustrates two ontologies used as global knowledge for specific domains of interests proposed in *Bibster* and *Semantic Overlay Network* architectures [44, 20].

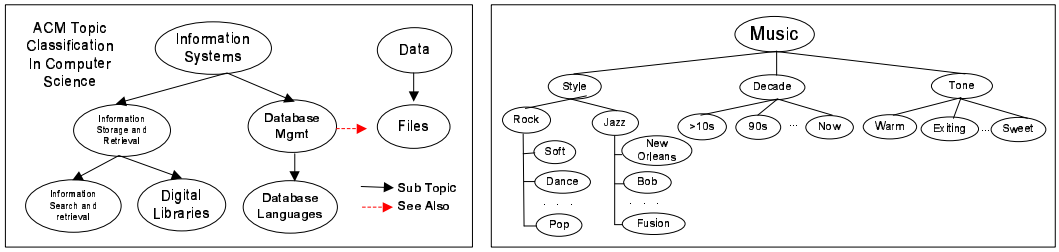

Figure 3: ACM and Music Ontologies [44, 20]

Despite its ease of deployment, this approach provides quite a restricted use of ontologies. Everyone is required to agree on the meanings, concepts, and relations defined in the ontologies. An approach based on global knowledge can be used effectively for a group of users who share a specific domain of interests. However, when it is used in a larger user group, where users have various domains of interests, this approach seems impractical. One may argue that concepts defined by users from a domain would not always be comprehended by the users from different domains.

Another approach uses concept mapping mechanisms provided by inference engines. The *P2P Semantic Web* (PSW) uses this approach to provide ontology interoperability by using a centralized server referred to as the *InfoQuilt*. The InfoQuilt server allows users to register, search, and reuse relevant ontologies in a P2P environment. PSW supports a Semantic Web language, namely *DAML+OIL* [59], which in turn enables users to define their own ontologies. By registering users' ontologies on the InfoQuilt, these ontologies are integrated into a global knowledge space. To resolve conflicts of ontologies, the InfoQuilt server automatically detects and avoids conflicts of ontologies when they are being registered. Users can also search for relevant ontologies to reuse or extend to their own, in this way the global knowledge is efficiently utilized and extensible to serve users' needs [8]. Despite the extensibility of global knowledge, the PSW has a short coming in its single point of failure since it relies on the InfoQuilt server. When the InfoQuilt fails, the system also fails.

Besides ontologies, *keyword relationships* are proposed to facilitate search in a *Semantic P2P Network* by expanding queries to include related keywords that each peer perceives. The keyword relationships are developed from the principle: "if a user issues a search with a keyword and selects a resource that is described with a another set of keywords, this user considers these keywords to have a relationship." Keyword relationships developed among peers are used as a taxonomy of semantics that are dynamic and reflect the community's interest [66]. Note that, although this knowledge is developed from the consensus of a group of users, it does not necessarily reflect an individual user's interest.

2.2.1.2 Mechanisms To Improve Search Efficiency

This section discusses the mechanisms to improve the search efficiency in a P2P environment. The mechanisms proposed

aim to either *reduce unnecessary search traffic* and/or *increase the scope of the search*. The mechanisms are classified into three categories: *random search*, *learning-based search*, and *semantic-cluster-based search*. The mechanisms in each category are discussed in the upcoming sections.

2.2.1.3 Random Search This approach targets mainly the reduction of the wasted bandwidth caused by blind flooding. The general idea is to randomly select only some neighbors to send search queries to within the network. The simplest mechanism of this approach selects a fixed proportion of forwarding neighbors in each step of a search [51]. A variant of this approach referred to as the *Random Walk* approach uses a fixed number of queries, referred to as walkers, to propagate independently within the network. Using a random walk, a peer starts by sending a set of walkers. Each walker then creates its own path by randomly selecting the next hop from peer to peer. A walker stops when it finds the resource or its time to live has expired. Results of using random walks show that the number of messages is reduced by more than an order of magnitude when compared to the blind flooding scheme [56]. This approach also provides load balancing properties, since messages are forwarded uniformly among neighbors. However, the performances vary, since the success of a search depends on the search starting point and the network topology that changes over time [28].

2.2.1.4 Learning-based Search The learning-based search uses the history of searches to optimize search performance. Each peer watches search activities of other peers through queries that are sent through it in order to develop its *local knowledge base*. When the peer wants to search for a resource, the peer uses the knowledge base to decide which neighbors it should sent its queries. The neighbors that have been involved in similar searches are selected, since these peers are likely to have acquired the answers in the past. In this way, blind searches can be avoided and the search can be more effective.

Adaptive Probabilistic Search (APS) improves *random walks* by using a probabilistic approach based on the responses from searches. Each peer maintains a local index consisting of one entry for each object it has requested per neighbor. The value of this entry reflects

the relative probability that this neighbor will be chosen as the next hop in a future request. When starting a search, a peer selects neighbors to send queries to based on their associated probabilities. When a walker successfully completes a search, it returns to the searching peer using the same path, and increases the value of the entry for the neighbor associated with that object. Using this mechanism, the values associated with the neighbors are adjusted to reflect the success rates of using them to forward the requests. By using these values as a factor to select the neighbor peers to search, the success rate of a search in the network is improved [95].

Neurogrid improves search efficiency by using the local knowledge base which associates a keyword search with neighbors that have queried or answered similar keyword searches in the past. Neurogrid was developed from the principle that "the peer that has sent queries and answers queries associated with a set of keywords will likely have resources associated with this set of keywords." Using Neurogrid to search for a resource, a peer first consults the knowledge base to find neighbors that are associated with the keywords specified in the search. If the peer has neighbors associated with these keywords, it then forwards the query only to these neighbors. Otherwise, the peer triggers a blind search to continue the discovery [50]. An overview of a search in Neurogrid is illustrated in Figure 4.

Similar to the mechanisms used by Neurogrid, the *Interest-based Shortcut* develops the knowledge base from the principle: "if a peer has a particular piece of content that one is interested in, then it is likely that it will have other pieces of content that one is also interested in" [89]. The interest-based shortcut approach suggested that when a searching peer finds an object in another peer, it should create a shortcut to that peer. This peer can be its local (close in distance) or remote neighbor. Later, if the searching peer wants to find a resource, it sends a query to all its shortcuts, even though they do not seem to have similar resources. If no object is found, it rolls back to use a blind search. The interest-based shortcut experiments showed that the Interest-based locality existed and was inherent in many content sharing workload. Additionally, it was effective at locating contents in P2P systems [89]. Figure 5 depicts an example of an interest-based shortcut.

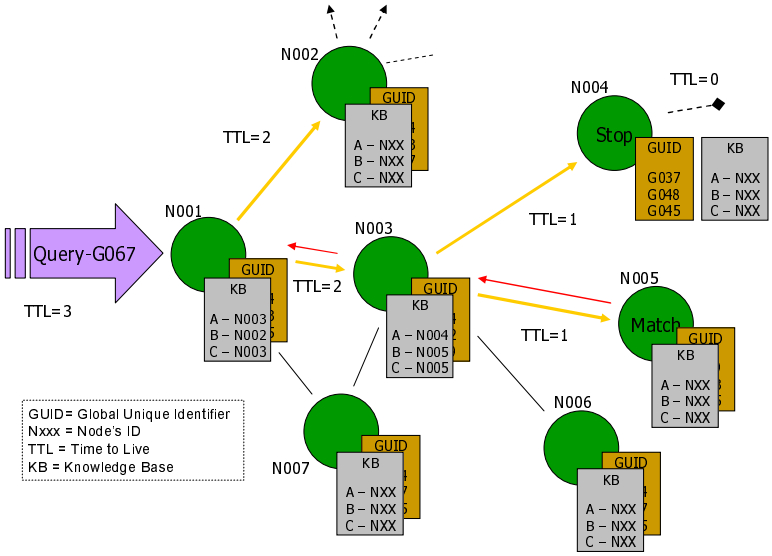

Figure 4: Search in Neurogrid.

Instead of using search history, the *Semantic P2P network* approach develops the knowledge base from search feedback. This feedback is used to adaptively adjust keyword relationships in the local knowledge base. Peers in the semantic P2P network represent their resources with sets of keywords. Each peer maintains a knowledge of keyword relationships initially setup by its local resources and dynamically developed from feedback from other peers. When a peer performs a search using a keyword, other similar keywords extracted from its knowledge base are automatically attached. In this way, a search can find resources which are described by similar keywords known to neighbor peers [66].

2.2.1.5 Semantic-cluster-based Search The semantic-cluster-based approach uses the concept of the *semantic overlay network* in which peers are connected to other peers based on the semantic similarities of their resources. It is believed to be an effective way to share resources in a P2P environment due to two data mining research premises. The first premise claims that the locality of hosting resources is highly coupled with the search. In other words, users are likely to search for things that are similar to what they have. The second premise specifies that users are likely to find resources they are looking for among users who share

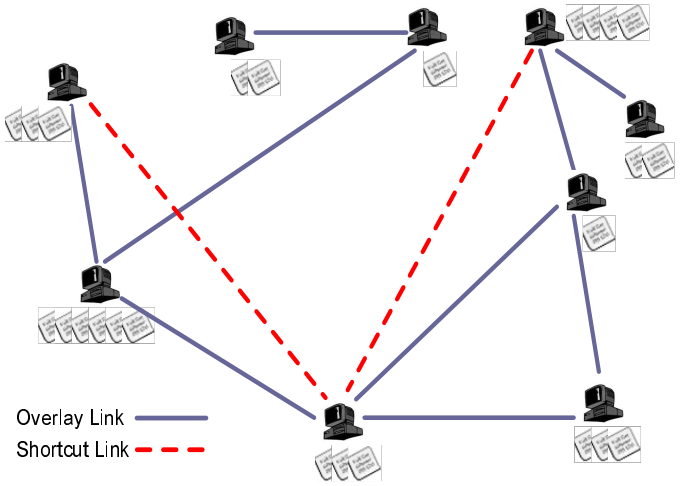

Figure 5: Interest-based Shortcut.

similar content [18, 46, 24, 25]. Based on these two premises, finding resources in a *semantic overlay network* is more effective and efficient than the previously proposed approaches.

The *Semantic Overlay Network* was first introduced in a music-file-sharing system. In this system, peers used the same music classification (Figure 3) to form semantic overlay networks. A peer participated in the semantic overlay network that shared music files in the same categories. To search for music files, a searching peer first built a list of semantic overlays to which its query would be sent. It then selected peers that belonged to these overlays and sent queries to them. Each receiving peer then tried to find the matches in its local files and returned them to the searching peer, or forwarded the query to other peers that were in the same semantic overlay network [20]. Even though this work provided a fundamental framework for using a semantic overlay network to improve resource discovery and sharing, it left many issues unaddressed including how to find the semantic overlay network specific to a user's interest, how to route within the semantic overlay network, and how to handle the dynamism of contents shared among peers.

Bibster is an interesting implementation that addresses the issues of implementing a semantic overlay network. Bibster is a P2P system for exchanging bibliographic data among computer scientists. Its goal is to allow users to discover and share the bibliographic data

which is kept in BibteX format. Bibster uses Semantic Web technologies, namely the Resource Description Framework (RDF), and ontologies provided by the ACM topic hierarchy as illustrated in Figure 3 to describe resources in the system. Its underlying infrastructure is an unstructured P2P network that uses the JXTA[5] communication platform. Bibster's semantic overlay networks form based on the semantic classification of the ACM topic hierarchy. Each peer is represented semantically by a set of ACM topics to which its content belongs referred to as an *expertise*. The peers that have similar expertise connect to each other to form a semantic overlay network. Through advertising, a peer informs other peers about its expertise and discovers other peers that have similar expertise. The informed peers may connect to the advertising peer if they have similar expertise, otherwise it forwards the advertisement to its neighbors. By using this semantic overlay, a search query is guided to propagate among the peers that have *expertise* similar to the searched topics. Each receiving peer computes similarity scores between topics in the query and the expertise of the other peers it knows about. The peer returns a list of similar peers ranked by the similarity scores to the searching peer. The searching peer then visits these peers to retrieve their resources [44].

Having addressed the issues regarding Bibster, it is important to examine the possibility created by the *Associative search*. The associative search approach uses the concept of *guide rule* to define a semantic overlay network in which peers satisfy some predicates such as the possession of the files. To perform a search using the associative search method, a searching peer selects the related predicates to find a group of peers to which to send the message. These peers then continue to forward the message within the guide rule until the search conditions, such as number of found resources or number of forwarding hops, are satisfied. The associative search exploits the association inherent in human selection with a basic premise: "peers that would have been able to satisfy previous queries by the originating peer are the more likely candidates to answer the current query." By performing a flood search only to a group of peers that satisfy the guide rule, the search then is forwarded to only a group of peers that are likely to answer. The basic predicate which identifies the

[5]A set of open protocols based on Java technology that allow any connected device on the network to communicate and collaborate in a P2P manner. Project URL: http://www.jxta.org/.

possession of the object is called the *possession rule*. The result from the simulation of a search using the possession rule shows that it does not only reduce bandwidth consumption, but also increases the scope of the search [19]. Yet, it is not clear either how the guide rule is developed among peers or how the system can handle a large group of peers. Without a mechanism to allow a peer to participate in a guide rule effectively, it is possible that a peer with items that satisfy the predicate for a guide rule may not participate in that rule. The resources in these peers, therefore, will not be found by a search which satisfies that rule.

Having reviewed the search mechanisms in unstructured P2P overlay networks, one can conclude that, even though these mechanisms can be used to improve the search performance of the Personalized Web, many issues related to scalability and lack of the search guarantee will need be addressed in the future. That is, when the number of peers in the network grows as large as a billion peers, finding an existing resource to develop a Personalized Web takes a variety numbers of discovering steps. It may be too costly in the number of steps it takes for discovery to happen, for instance, in the worst case a billion steps. It also maybe a few steps when a query is sent to a peer that has those resources. The next section describes structured P2P overlay networks and the search mechanisms which have been proved relatively more scalable and robust and provide a search guarantee.

2.3 STRUCTURED P2P OVERLAY NETWORKS

Structured P2P overlay networks have recently gained attention as a viable solution to facilitate resource discovery and sharing in large scale networks. The foundation of this technology is a distributed hash table (DHT), which is used to infer a structure that regulates location and access to resources distributed among peers. As results of the DHT inferred structure, overlay peers randomly and equally maintain resource information on the network. Furthermore, structured DHT-based P2P overlay networks provide a search mechanism that is scalable, and adaptive. Using the DHT information, overlay peers, which currently manage the resource, can be identified deterministically and in a finite number of steps. Typically, resources can be located in a number of steps in logarithmic order to the number of peers in the network.

The DHT-based overlay architecture is used by several overlay network substrates, including Tapestry, Pastry, Content Addressable Network (CAN), and Chord [106, 78, 73, 90]. These substrates present different frameworks to use distributed hash tables, yet they share the fundamental algorithms for routing, search, data location, and network formulation. To explain these algorithms, a general framework of a DHT overlay network is summarized as follows:

- Each peer in a DHT-based network is assigned an unique identifier (ID) generated by a system-wide hash function which locally applies its unique attribute, such as the IP address or the peer's public key.
- Each object in this network is also assigned an ID generated by a system-wide hash function that applies to a resource attribute, such as filename, or the file itself.
- Each peer is assigned the responsibility for managing objects that have IDs in its range, that is, the range of IDs that are equivalent or closest to the peer ID.
- Each peer also maintains a routing table containing routing matrices to the peers that are selected by some criteria, such as peer ID distance, a routing algorithm, and a neighboring policy.
- Any objects or queries that arrive at the network are routed towards the peer IDs that are equivalent or closest to the objects' or queries' IDs (these peers are referred to as the targeted peers for the queries).
- Routing uses only local forwarding, a decision made by each peer along the path to the targeted peer.
- The routes maintained by a peer are classified into several levels of search space granularity, such that the routing in each step takes a smaller discovery distance from the destination (a routing table of a Pastry node that contains several levels of search space granularity is shown in Figure 6).
- Each DHT substrate uses a group of rendezvous peers to provide initial information of the network to allow a new peer to join the network. To join a network, a new entering peer first sends joining request associated with its peer ID to a rendezvous peer. The rendezvous peer then routes the request toward this peer ID. The peers receives the

request along the path then send back the information of the network that it knows of to the joining peer. The joining peer uses this information to setup its initial state. The peer continues to negotiate and inform neighbors about its logical position in the overlay network until all neighbors are discovered and accept its position.

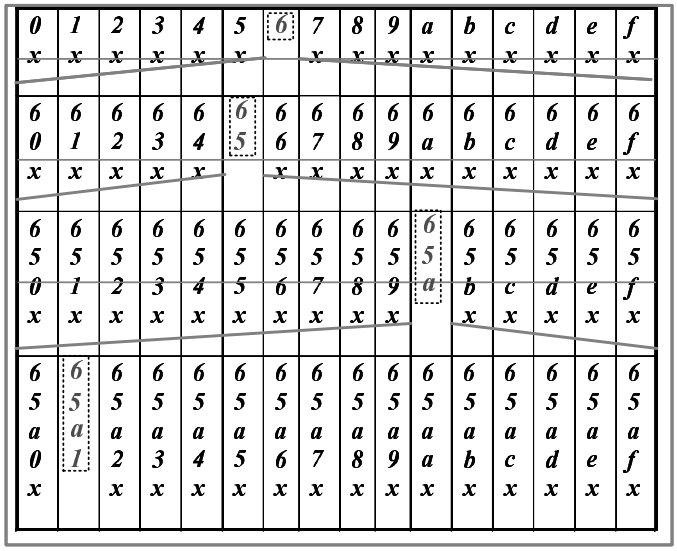

Figure 6: A Pastry Routing Table.

To further explain routing in a DHT network, an example of sending a query associated with ID *d46a1c* from a peer with ID *65a1fc* to a target peer using a Pastry substrate is described below and illustrated in Figure 7.

1) Peer *65a1fc* first forwards the query to *d13da3*, which is one of its maintained routes, and which is closest to the query ID *d46a1c* because it shares the first digit of the object ID, i.e. *d*. Note that the routes a Pastry peer maintains have different levels of shared key spaces (granularity) based on its peer ID. For example, the route that peer *65a1fc* maintains are classified into six groups: the group of peers that has a different first digit ID, the group of peers with the same first digit, i.e. *6*, the group of peers with the same first two digits, i.e. *65*, ..., the group of peers with the same first five digits ID, i.e. *65a1f*. This routing table is shown in Figure 6. Based on the

peers' IDs in the routing table, each routing peer selects the next peer that has the longest digit matched to the target ID to forward the message to.

2) Peer *d13da3* forwards to peer *d4213f*, since it is one of peer *d13da3*'s neighbor nodes and it shares two digits with the target objects ID, i.e. *d4*.

3) The forwarding continues as shown in the Figure until the message reaches the peer *d467c4*, which has the ID closest numerically to the target key (*d46a1c*).

Note that the Figure shows a circular representation with peer IDs increasing clockwise. These peer IDs do not necessary correlate to their physical locations.

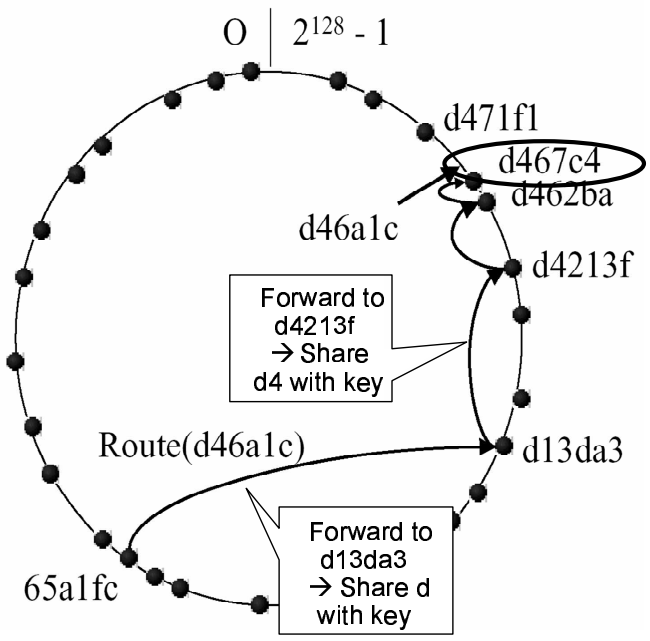

Figure 7: Routing in Pastry.

Based on the provided algorithms used in the example above, the keys to the scalability and robustness of the routing scheme can be summarized as follows:

I) The number of discovering steps is bounded by $log_b n$, and each peer maintains no more than $b * log_b n$ routes, where n is the number of peers in the overlay and b is the digit base representing the ID.

II) The network is self organized by each peer that loosely collaborates with others to allow peers to freely join and leave network without interrupting resource discovery and sharing.

III) The maintenance costs in terms of CPU cycles, storage, and bandwidth are shared among all the joining peers.

The other P2P overlay substrates, including Chord, CAN, and Tapestry, also give similar performance matrices. Table 1 displays a comparison of the performance metrics for these substrates.

Table 1: Comparison of the Structured P2P Overlay Substrates [62]

P2P systems	**Model**	**Parameters**	**Logical path length**	**Neighbor state**
Chord	Uni-dimension circular ID space	Number of peers in the network – n	$log_2 n$	$log_2 n$
Can	Multidimentional ID space	Number of peers in the network – n Number of dimensions – d	$d * n^{1/d}$	2*d
Tapestry	Global mesh Plaxton style [71]	Number of peers in the network – n Base of chosen identifier – b	$log_b n$	$b * log_b n$
Pastry	Global mesh Plaxton style [71]	Number of peers in the network – n Base of chosen identifier – b	$log_b n$	$b * log_b n$

2.3.1 Resource Discovery in Structured P2P Overlay Networks

The properties of DHT overlay networks make them a viable solution to support resource discovery and enable management of resources in a distributed environment. However, the basic DHT-based overlay structure exhibits several shortcomings that need to be addressed in order to meet the performance requirements of resource discovery.

One critical limitation of the DHT-based structure with respect to resource discovery is its lack of support for locality-based queries that go beyond perfect matching. Resource distribution in a structured P2P overlay network is based on a hash table which maps an item to a single key. This key is mapped to one peer in the overlay network. As a result, each resource is uniquely handled by a specific peer in the network. This resource indexing scheme does not support searches based on semantics. For example, searching for a paper with keywords "car engine" will not find the resources indexed with keywords "automobile engine." One could index resources with multiple related keys (i.e. both car engine and automobile engine). By using semantic keys in the hashing scheme, the random property of resource distribution in a DHT network is compromised. Different semantic keys will likely be associated with a different number of resources. Similar to their functions in web indexing systems, semantics, such as keywords appearing in queries or that are associated with resources, are used in a non-random fashion [12, 21, 5]. Some popular keywords are likely to be used frequently, while some keywords may not be used at all. Consequently, when partitioning resources based on semantic keys in DHT-based network, some peers that are responsible for popular semantics must handle the high traffic load caused by a large number of queries and indexing. At the same time the peers that are responsible for rarely used semantics will receive only a few queries or nothing at all. A DHT-based resource discovery system must address this issue by providing mechanisms to distribute the servicing load and storage in an effective and scalable manner.

The following sections discuss the mechanisms proposed to address the issues above. The discussion is divided into three subsections: the keyword-based approach, the metadata-based approach, and the ontology-based approach.

2.3.1.1 The Keyword-based DHT Approach

The keyword-based DHT approach, as its name suggests, uses keywords to associate data with nodes in a DHT overlay network. Each peer in the network is responsible for the data associated with keywords that are mapped to it. Several schemes were proposed to provide key-mapping to the peers. The simplest scheme maps each keyword to a peer in the network [76]. Using this scheme, a peer performing a search first computes the keys for the keywords specified in the query using a

hash function. The generated keys are then used to route the query to the peers responsible for them using the DHT strategy described in the previous section. Once the query reaches these peers, they then return a list of all resources associated with the keywords.

Despite its simplicity, this scheme requires a potentially large amount of data to be transmitted for a multi-keyword search. For example, a resource associated with three different keywords will be indexed by three different peers. When issuing a search associated with these keywords, three queries will be routed to these three peers and all of these peers then return results to the requesting peer.

A set of keywords referred to as *keyword set* (KSS) is proposed to increase efficiency in a keyword-based search. Instead of using a single keyword, KSSs are associated with resources and indexed among peers [34]. In this way, a search with multiple keywords generates a smaller number of queries and a smaller number of returned lists compared to a single keyword scheme. The effectiveness of this scheme, however, depends on how the KSSs are formulated and used in the search queries.

The study on the feasibility of using the aforementioned keyword-based schemes argued that using these techniques to support web indexing and search was impossible because of the communication and storage constraints. To make it feasible, researchers suggested the use of optimization techniques such as *Bloom Filters*[34, 63], *Gap Compression*[103], *Adaptive Set Intersection*[26] and *clustering*[47]. The main idea was to reduce the amount of data that needed to be transferred and stored by using the pre-processing techniques mentioned above[54].

pSearch was proposed to improve resource indexing in P2P information retrieval systems using keyword vector representation. The keyword vector representation is supposed to be a better choice for representing resources and queries than a plain keyword-based representation because it can represent not only the presence of keywords, but also their importance, often referred to as weights [92, 91]. In addition to using keyword vectors to represent and index resources and queries, *pSearch* employs certain IR techniques, namely the *Vector Space Model* (VSM) and *Latent Semantic Indexing* (LSI) with an underlying CAN-based DHT network. VSM is a semantic model that adjusts weights of terms with the uniqueness of the terms in the index, while LSI is a technique that extracts weights of terms using co-occurrence

of terms in documents in the index. They both can be used to draw semantic importance in the form of vector representation. Typically the similarity between a document and a query is measured by the cosine of the angle of their vector representations. These techniques are integrated into CAN to form a *semantic overlay network* in which peers share semantically similar contents. A *semantic overlay network* is used to map resource indices among peers and to provide semantic-based routing for searches in the system.

Using *pSearch* to advertise a document in a CAN network is similar to placing it into its semantic position in the semantic network. A peer first calculates the semantic positions of the resource using VSM or LSI. The keys of semantic positions are then computed using a hash function. The keys are then used to locate the peers that maintain the CAN-zone covering these positions. Finally, data is routed to the targeted peers using the CAN routing strategy, similar to the Pastry routing strategy described in the previous section.

A search using *pSearch* scans both the semantic positions mapped to the keyword vectors as well as the nearby positions where similar contents are stored. The routing part of the search is similar to the advertising. First, the semantic positions of the query are calculated and then the query is routed to the targeted peers that are responsible for these semantic positions. However, when the query reaches these peers, the query is flooded to peers within a radius r, determined by a similarity threshold or a number of retrieved documents specified by the user. In return, this mechanism increases the scope of the search and allows similar resources to be retrieved. Figure 8 illustrates a simple scenario of a search using *pSearch*.

The *Hilbert Space Filling Curve(SFC) DHT architecture* uses a keyword vector coupled with an SFC over a Chord-based DHT network to provide distributed web service discovery. Technically, an SFC is a continuous mapping function that transforms a d-dimensional space to a *1*-dimensional space $f{:}N^d \rightarrow N$ [82, 83]. The d-dimensional space is viewed as a d-dimensional cube, which is mapped onto a line such that the line passes once through each point in the volume of the cube. Each point along the line represents a d-dimensional coordinate suitable to describe the characteristics of resources. Figure 9 shows an example of an SFC for 2 dimensions with two and four parameters.

The SFC DHT architecture uses SFC to map keyword vectors representing web services to 1-dimensional addresses. Placing or advertising a resource on the network is achieved

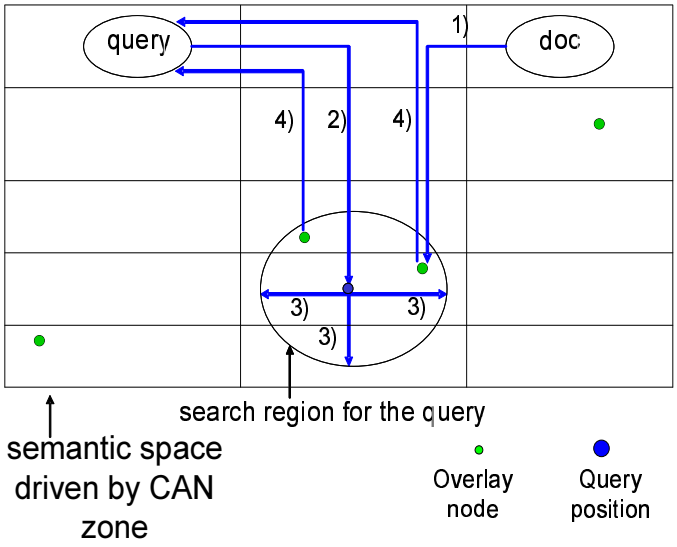

Figure 8: Search and Query using pSearch.

in three steps. First, a peer obtains the hash value of the SFC point that matches the coordinates of the keyword vector describing the characteristics of the desired services. The peer then routes the resource metadata to the peer that is responsible for the hash value of the SFC point. Finally, when the metadata arrives at the targeted peer, the targeted peer adds the resource metadata to its index. Figure 10 illustrates the advertising process of the SFC DHT architecture.

Querying for resources using SFC involves similar processes. For instance, the peer first obtains SFC point(s) for the query. It then sends the query to the peers responsible for the hash values of these points. Figure 11 illustrates a querying process in this architecture.

pSearch and *SFC DHT architecture* also provide the mechanisms to deal with one of the main problems in resource indexing in a DHT overlay network: an unbalanced index and load distribution. One mechanism employs newly joining peers to help carry the load from existing, overloaded peers. To illustrate, when a peer enters the network, it is given alternatives for its peer ID. This mechanism suggests that the joining peer selects the ID such that it is close to the peers carrying heavy loads. In this way, the load is distributed among new peers. The more peers join, the better the load distribution within the network

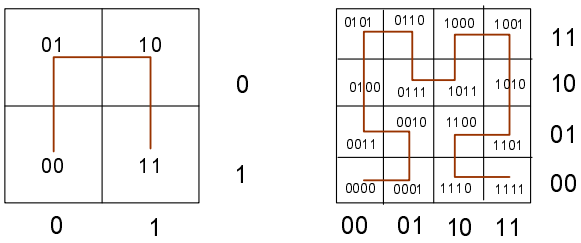

Figure 9: A Space Filling Curve for 2-Dimensions with 2 and 4 Parameters[82, 83].

will be. In addition to using the entering peers, *SFC DHT architecture* exploits the load migration techniques that are used at runtime. Each peer periodically exchanges its load information with its neighbors. When the load of a peer is greater than those of its neighbors by a specified degree, the load of that peer is distributed to its neighbors.

2.3.1.2 The Metadata-based DHT Approach Metadata in the form of a set of attributes and values have been increasingly used to represent resources on the Internet because of the ability to describe various resource semantics specific to users' needs. A metadata is typically used to describe resource semantics that keywords alone cannot represent. For example, the semantics of "a PDA device that has wireless card" and "a sun station that has 10 gigabyte storage space" can be represented by metadata (device=PDA,property:wireless card=true) and (machine=sun station, harddisk-size=10Gb). Because of its extensibility and expressiveness, interest in metadata in resource indexing within a DHT based overlay network is growing.

Intentional Resource Discovery (IRD) architecture uses metadata to represent resources' and queries' semantics in order to allow peers to locate services and devices in a large scale environment. IRD is comprised of two main components: *INS* and *Twine* [9]. *INS* was originally designed as a new naming system intended to support naming and discovering various resources on the network. *Twine* is the core routing and data location infrastructure working on top of a Chord overlay network. *Twine* deploys a group of dedicated peers called "*resolvers*" to allow services to advertise their presences. *Twine* also allows clients to search

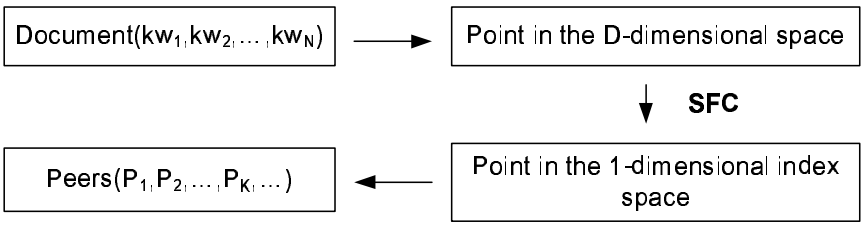

Figure 10: Advertising Using SFC [83].

for services in the network. These resolvers form a Chord-based overlay network allowing the advertisements and searches to effectively route among peers in the network. Resource advertisements and search queries in IRD are represented by semi-structured descriptions. Each description includes the current IP address, port number, and protocol description of the resource being provided. In order to increase the scope of the discovery and locality of the search, the IRD divides resource descriptions into small, redundant pieces referred to as the "strands." *Strands* of a resource description are mapped to the resolvers responsible for their hashed keys. In this way, a search for a subset of metadata's attributes can find the resource. Figure 12 illustrates an example of advertising a resource using *INS/Twine*.

In order to improve the effectiveness of a search, IRD extracts a set of *strands* from the query to generate an efficient search sequence. To retrieve resources for a query, first the longest *strand* is sent to the resolver responsible for its hashed value. If the receiving resolvers find a matching resource, they return the resource description to the requesting peer. The search process continues with a shorter *strand* until either the peer is satisfied with the returned results or there is no *strand* left to discover. Figure 13 describes a search example corresponding to the advertising example described above.

The *Augment P2P data indexing (APDI)* system is an implementation that utilizes emerging structured language technologies, namely XML and XPath [100] to support partial matching in the DHT network. The APDI provides key-to-key mapping which allows peers to search for keys referenced to data [39]. A resource description is divided into partial descriptions each contains a subset of the attribute and value pairs of the original description.

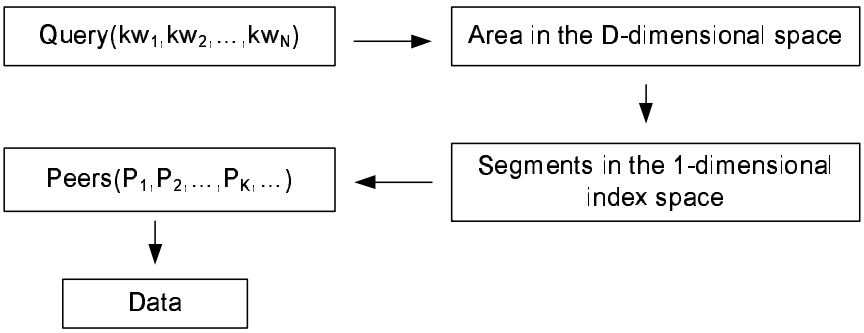

Figure 11: Querying Using SFC [83].

Each partial description associated with the reference of the original description is sent to the peer responsible for that part. In this way, a search query that matches a part of a resource description can then find the resource through its reference.

Advertising a resource using APDI involves handling indexing partial descriptions in a DHT network. A resource description is first divided into a set of partial descriptions. Then, the resource is sent to be stored at the peer responsible for the original description, while the references are sent to be stored at all the peers responsible for its partial descriptions. Figure 14 illustrates advertising of a resource f with a descriptor d and a set of partial descriptions $Q = q_1, q_2, q_3$. The references $d : f$, $q_1 : d$, $q_2 : d$, and $q_3 : d$ at peers $h(d)$, $h(q_1)$,$h(q_2)$, and $h(q_3)$ are stored at the peers responsible for its partial descriptions.

The search process of APDI is similar to that used in IRD, except the return result can be a data reference, which requires an extra redirection, in APDI. The original description is sent to find resources. If a match to the original description is found, the resource itself is returned. If the search does not find any match, the search continues with an untried, longest partial description until the search criteria are satisfied. On the other hand, a match with partial description results in receiving a reference to the original description. Once receiving a reference, the search redirects the query to the peer responsible for the original description in order to obtain the resource. Figure 14 illustrates the way a user queries a resource using APDI. First, the user sends query q_1 to peer $h(q_1)$. The peer $h(q_1)$ then returns the reference to its original description, i.e. d. The query is then sent to peer $h(d)$ to obtain resource f.

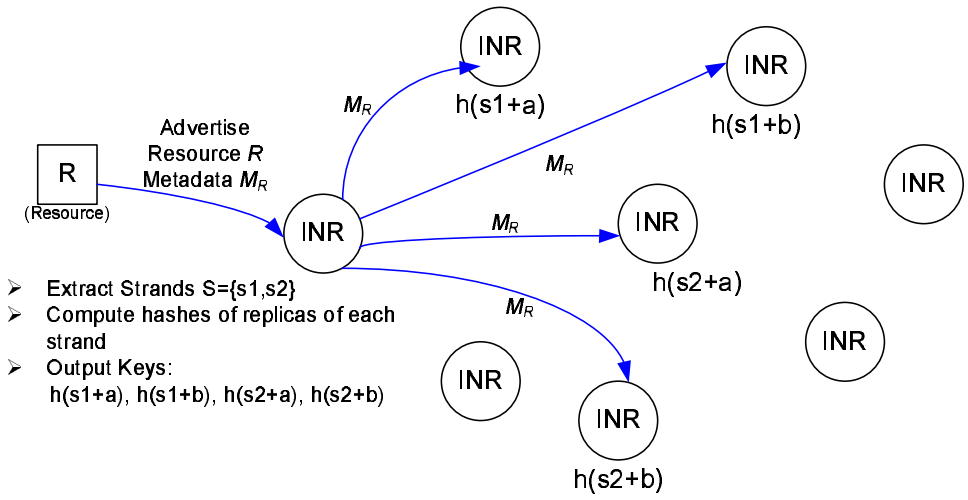

Figure 12: Advertising using *INS/Twine*.

2.3.1.3 The Ontology-based DHT Approach The ontology-based DHT approach partitions a resource index among peers based on a taxonomy defined by an ontology. It is relatively new, compared to the keyword-based and the metadata-based DHT approaches. However, it has gained a lot of attention in recent years because of the benefits of the expressiveness and the self-described semantics provided by ontology. The ontology allows a DHT-based system to support semantically based searches which go beyond word form matching. For instance, a search for the concept "car" can be automatically extended to include "automobile" so that the query is sent to both peers responsible for these concepts. Similarly, a search with the concept "network security" could trigger the sending of a query to the peers responsible for a set of related concepts including "network authentication," "encryption," and "denial of service attack."

Taxonomy-based Routing in P2P Networks employs a global catalog taxonomy to partition contents in DHT-based networks. A global catalog taxonomy defines a hierarchy of categories of a domain of interest which are used to represent the semantics of resources [55]. A hierarchy path identifies relations of categories in the path. A resource assigned to a set of categories is described by the associated category paths. The paths are included in the resource description indexed in a DHT network. The possible prefixes of the paths are also

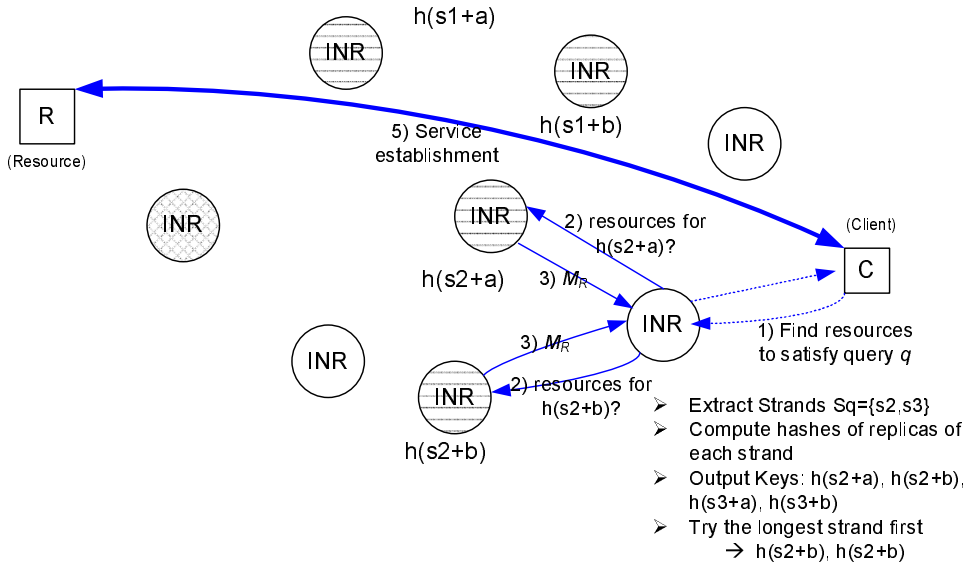

Figure 13: Querying using *INS/Twine*.

stored in a DHT in order to increase the scope of the search. This extra storage in turn makes it possible to perform queries both on the exact path as well as on the related paths defined by the taxonomy. This way, searching for resources in a category could acquire not only the resources defined in that category but also those in the categories under it. For example, when the resources in the *Programming* category are requested, resources in successor categories, like *Java* and *C++*, are also requested. By placing category paths in a DHT, peers can forward the request to the peers responsible for the related categories immediately.

The global catalog DHT architecture also exercises a set of storage load balancing strategies to alleviate unbalanced load and index distribution at the cost of an extra level of redirection. It uses a set of *load directories* to recommend overloaded peers, which they can use to help carry loads for them. These load directories maintain the load information of peers that have not reached their limits. Peers periodically advertise their current and target loads to one of load directories. When a peer becomes overloaded and is requested to add data, it send a request to one of these load directories. The receiving load directory then randomly selects a peer that has a target load higher than the current load and returns the peer's

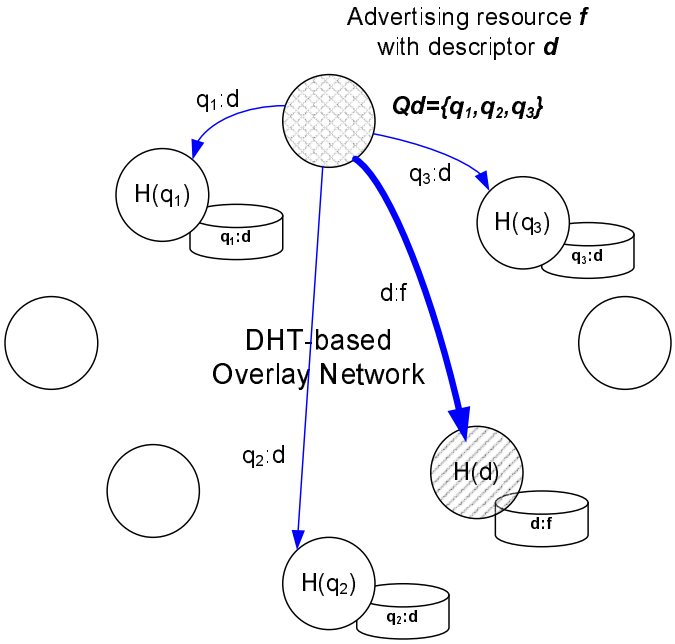

Figure 14: Advertising in the Augmented P2P Indexing System.

address to the requesting peer. The overloaded peer then redirects the newly added data to the selected peer and records the address of the selected peer associated with the requested category. In this way, when data from this category is requested, this peer can collect all the data from the redirected peers.

Having reviewed mechanisms to improve resource discovery in both unstructured and structured P2P overlay networks, three main conclusions with regards to resource discovery in a P2P environment can be made. First, ontology-based representation possibly could be used to represent resources and queries in order to improve semantically-based resource discovery. This conclusion can be drawn from the fact that the semantic representations in the form of keywords and metadata do not address the synonymy and polysymy problems. Many proposed mechanisms try to address these problems by adding ontology capabilities including the ability to infer semantics from word relationships or feedback on top of the underlying keywords and metadata representations. Additionally, various literature have demonstrated the implementation of ontology-based resource discovery.

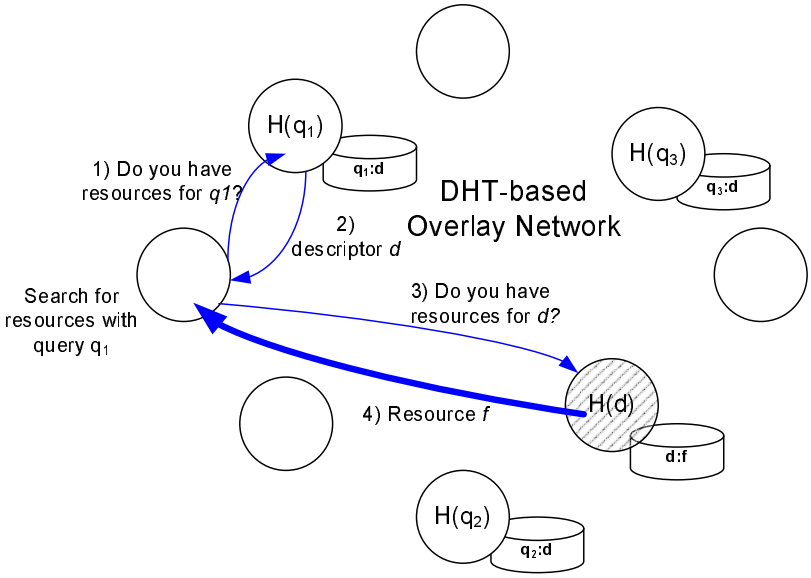

Figure 15: Querying in the Augmented P2P Indexing System.

A second conclusion could be drawn from the fact that the underlying search mechanism of the proposed approaches in unstructured P2P networks still depends on random or selective searches which do not guarantee that an existing resource could be found. Even though an unstructured P2P overlay network could implement the proposed mechanisms to improve the efficiency of resource discovery, it could not provide a discovery guarantee especially when the size of the network and the number of resources become very large. When the size of a P2P network becomes large, searching for a resource in the network may be hard to achieve within a certain scope. On the other hand, increasing the scope may be too costly to support.

The third conclusion suggested that the structured P2P network could be used to provide a scalable search as well as a certain search guarantee, yet one must provide effective mechanisms to support semantically-based indexing and search. Various literature demonstrated the scalability of resource discovery in DHT P2P networks. At the same time, the mechanisms to support a semantically-based search in these networks employ IR techniques namely Latent Semantic Indexing (LSI) and the Vector Space Model (VSM), or excessive

advertising which depends either on word-forms or on parts of metadata. One interesting approach uses a limited form of ontology provided by a category hierarchy taxonomy. The question of whether a DHT network could integrate ontology and support semantically-based resource discovery remains to be explored. The next section discusses the the underling network and the mechanisms that could be used to enable the Personalized Web in light of these conclusions.

2.4 USING A P2P OVERLAY ARCHITECTURE TO ENABLE THE PERSONALIZED WEB

Having reviewed the literature that provided resource discovery, access, and sharing using P2P overlay technologies in the previous section, this section discusses the capabilities of these mechanisms to support basic functionalities of the Personalized Web which include:

- resource discovery based on user specific profiles that are associated with ontologies of interest in a user transparent manner,
- user-selective resource advertising among user communities of similar interests, and
- the dynamic formation of adhoc interest groups by recruiting new members with a similar interest.

The purpose is to answer the question of whether or not the architectures and mechanisms proposed by the literature can address the current problems with the design and development of the Personalized Web. Based on the above functionalities, the following sections explore this question by investigating the capabilities of the mechanisms that support resource discovery, resource advertising, and the formation of communities of interests.

2.4.1 Resource Discovery

One of the main objectives of a Personalized Web is to develop a personal web of interest that contains resources specific to the user's interest in a user-transparent manner. To achieve this objective, Personalized Web architecture must enable automated processes to discover

resources based on semantics specific to the users' domains of interests. The architecture also must allow these processes to use users' similarity profiles to develop a personalized structure representing the semantic distances between discovered resources and the users' resources. Furthermore, the Personalized Web architecture must be able to support these mechanisms in the Internet environment where the numbers of resources and users are growing quickly and their presences change over time. The mechanisms, protocols, and data structures that enable resource discovery must be scalable and robust.

Having reviewed the resource discovery in P2P networks, the proposed mechanisms alone cannot be used to fully support an automated process to discover resources based on the user's interests. Most of the proposed mechanisms aimed at facilitating human searches which are mostly based on word-forms could not be used to support this functionality. These proposed mechanisms provide a way to find only resources that are associated with the search word-forms, they do not allow any automated processes to infer different semantics from word-forms. In order to enable this functionality, the Personalized Web architecture must employ explicit, self-described semantics to describe resources and queries that express user interests in a consistent manner. In this way, automated processes can find resources that match the user's interests using resources' semantics and the user interest profile in an effective manner. Two main approaches that could be used to guide automated resource discovery include Bibster and Taxonomy-based Routing. While Bibster could be used to allow an automated process to draw keyword relationships used in search, Taxonomy-based Routing could be employed to guide automated processes to search categories extracted from the taxonomy paths. However, these two approaches provide a limited form of semantics and ignore other semantic relationships such as the "is-a-part-of","belongs-to", etc. which could significantly improve automated resource discovery.

Most of the proposed mechanisms could not be used to support resource discovery driven by users. All of the literature proposed global mechanisms to discover resources among peers based on system similarity metrics. Only two of these works, namely *Associative Search* and *Interest-based Shortcut*, allow users to use their own semantics for discovery [19, 89]. The concepts of the *guide rule* and *interest-based locality* can allow peers to define their own

interests and to acquire resources or peers of interest. However, since both of these concepts are used in unstructured P2P networks, the scopes of the discovery are limited to their overlay topologies.

Considering the scalability and robustness of resource discovery of the Personalized Web, one must choose either unstructured or structured P2P network to be its underlying P2P overlay network. On the one hand, using an underlying unstructured P2P network could provide a certain ease of deployment and better support for richer semantically-based queries. However, such an approach could not provide a search guarantee and it could be inefficient and it may not support resource discovery in a large scale network. On the other hand, using an underlying structured P2P network could provide a highly scalable search, data location and a search guarantee. Yet, in order to support the rich semantic search in the underlying structured P2P network, it requires various mechanisms to address the limitations of perfect mapping and load balancing among peers.

Aimed at providing scalability and robustness resource discovery, the Personalized Web will employ an underlying structured P2P network. Using a structured P2P overlay network could allow automated processes to route, locate data and find members who share a similar interest quickly, consistently, and efficiently. The benefits outweigh the complexities of dealing with perfect mapping and load balancing issues. Load balancing and perfect mapping can be distributed to be handled by the peers in the network. The challenge of using an underlying structured P2P network to support resource discovery for the Personalized Web will be addressed in the next chapter.

2.4.2 Resource Advertising

Personalized Web architecture must enable automated processes to advertise resources to the users who have interests. This advertising makes it possible for Personalized Webs to evolve through time in a user-transparent manner as the new resources become available to the users. The advertising mechanism expands resource discovery and sharing among Personalized Web users. Since they rely on resource retrieval only, the Personalized Web agents would not be aware of the newly available resources without periodically performing

discovery, which could be costly. Considering the mechanisms to disseminate information of resources in P2P networks, resource advertising might be implemented using a local knowledge base or a global assignment through a DHT scheme.

Resource advertising of Personalized Webs could be automated through the knowledge-base using mechanisms proposed by *Nerogrid*, *Associative Search*, and *Interest-based Shortcut*. Each automated process could build a knowledge-base that stores semantics associated with peers' locations based on the search queries it perceived in the past. When a process advertises a resource, the knowledge-base is then used to find targeted peers that use semantics relevant to the advertisement. This advertisement is then sent to these peers and, in turn, the automated processes on these peers continue to advertise to other peers based on their knowledge-base. Advertising continues until some criteria such as a number of forwarding hops is satisfied [50, 19, 89]. Despite its adaptiveness and robustness to the changing environment, this approach can only advertise within a limited scope. The target for an advertisement is completely dependent on the search history, which is setup in an adhoc fashion. Because of this dependency, the scope of advertising is limited to the network topology and search activities that each peer receives. It may be that peers interested in certain resources may not receive the advertisements since they are far away from the advertising peer and have not developed a search history with the advertising peer.

Resource advertising could employ the DHT resource indexing scheme to partition resource advertising among peers in the network [90, 73, 106, 78]. Each peer could act as a proxy to advertise for resources associated with the semantics that are mapped to it [55].

Multicasting DHT-based approach proposes a multicast infrastructure which could be used to support content-based advertising in a DHT network. The multicast infrastructure would allow peers to form a multicast group for notification of information that matches a topic of interest. To subscribe to a topic of interest, a peer could use the semantics of the advertising topics to route its subscription to the peer that acts as the root of the multicast tree [17]. If there exists any established path to the multicast tree, the subscribing peer could then connect along the routing path to the existing path of the tree. When a new resource

is advertised to a group, the advertising query is then routed to the root of the multicast group. In turn, the root of the multicast tree might propagate the advertisement to all the children it has for the group. The advertising could then continue until there is no child left to advertise to.

Even though they provide scalability and efficiency in query routing, these DHT-based advertising schemes incur a high cost for infrastructure setup and maintenance. Peers in the network must collaborate to setup a path for members in the multicast group. If one of these peers leaves the network, the multicast path needs to be repaired and re-established. Additionally, when members join and leave the group, the path has to be rearranged to cover the new members and exclude the departing ones. The advertising must be dynamic in response to the change of group members. In conclusion, the DHT multicast infrastructure could not be used alone to support the advertising infrastructure for a Personalized Web. A combination of the strategies used in unstructured and structured P2P overlay networks to support advertising in the Personalized Web must be considered.

2.4.3 The Formation of Communities of Interests (CoIs))

In order to enable users to efficiently expand and share their Personalized Webs with users who have similar interests, the Personalized Web architecture must allow agents to find other users who share similar interests, to form a logical network among them, and to facilitate searches among agents, advertise, and share access to the resources. This logical network is referred to as a Community of Interest (CoI).

A CoI could be formed through a semantic overlay network where a group of peers share predicates, such as the semantics of the resources [19, 20, 44]. A CoI could be considered to be a special type of semantic overlay network where the shared predicate is a peer interest, instead of the semantic relevance or resource possession suggested by the literature.

While the concept of the semantic overlay network was presented in the literature, it is not clear how a semantic overlay network where members join and leave dynamically could be discovered and formed in the Internet environment. The proposed semantic overlay

networks use search activities from the past to recognize the peers that share the same interests. However, since all of the literature uses an underlying unstructured P2P network, it would not be guaranteed that the history of searches could identify the peer's interests. Additionally, the proposed frameworks in the literature do not address how a peer in a semantic overlay network could expand its discovery based on other peers' findings. They simply use a flooding mechanism to forward advertising to all members within the semantic overlay network.

In order to allow CoIs to develop to reflect the users' interests closely, peers must be able to define their own CoIs independently from other peers. For example, a peer P_a may have peer P_b in its CoI, yet the P_b may not have P_a in its community. By enabling peers to develop their own CoIs, peers could efficiently retrieve and advertise their resources effectively. A peer could have an advertising CoI where the members are interested in its resources and another retrieval CoI where it discovers resources. The majority of the proposed frameworks do not realize these possibilities and mostly assume all members in a semantic overlay network always have the same interest. While this assumption could be true for a small semantic overlay network, it could hardly be used for a larger semantic overlay network due to the variety of interests between users.

2.5 CONCLUSION

This chapter has reviewed the literature that propose resource discovery using P2P overlay networks in order to answer the first thesis's question: *Is it possible to combine P2P overlay network and agent technologies to enable a Personalized Web?* The P2P overlay networks, their components and the mechanisms to provide resource discovery, access, and sharing are provided. Based on the typical classification, the discussion was divided into two categories: unstructured and structured P2P overlay networks. While the unstructured P2P networks provided a simple network formulation scheme with low maintenance costs, the structured P2P networks support highly scalable search and routing algorithms.

The weaknesses of unstructured P2P networks include the expensive blind-flooding discovery and the lack of search scope or ability to find rare items on the network. Three main approaches try to resolve these weaknesses: the random search approach, the learning-based search approach, and the semantic-cluster based search approach. The random search approach attempts to reduce the blind-flooding to selective-flooding based on some random variables. The learning-based search approach tries to improve search performance by allowing each peer to learn from search activities that occurred in the past, and decides accordingly which peers should it send the current query to. On the other hand, the semantic-cluster based search approach uses the concept of semantic overlay network to optimize the search through the overlay network. Its premise is that peers that share similar contents are likely to find interest data among them. By only flooding the semantic overlay network to which a peer belongs, a search is likely to be successful with less bandwidth wasted in the process.

For the structured P2P network, the main weaknesses include the ability to search based on flexible or partial match queries and the heavy protocols required to maintain structures in the network. These weaknesses exist because a structured P2P network relies on a distributed hash table (DHT) that provides only a perfect-mapping scheme, which requires extra semantic structures in order to enable semantically-based resource discovery. Based on the semantic structures used to partition resources indices among peers, the approaches used to enable semantically-based resource discovery in a structured P2P network are classified into three categories: keyword-based, metadata-based, and ontology-based. As the names suggest, keyword, metadata, and ontology each are used in these architectures respectively. They are used to map data responsibilities to peers in the DHT networks. To index or search for data, these semantic representations are used as semantic keys to locate the peers that are responsible for them.

In addition to dealing with perfect mapping, a DHT-based architecture must handle both unbalanced storage and index processing load among peers. Due to the fact that each semantic is not used equally to represent resources and queries, a peer may deal with a lot of resources and queries or may rarely find one to handle. Because of this unbalanced indexing and storage load, the peers responsible for popular semantics could be overloaded.

To address this issue, various techniques, ranging from using a random subsequence of strings to the employment of load migration between peers were proposed.

Having reviewed the literature in terms of resource discovery in a P2P environment, one can conclude that using unstructured P2P networks as the underlying network architecture to provide semantically-based resource discovery would provide better support of semantics used in search and indexing schemes with a lower search scope and a lower cost of maintenance. On the other hand, using an underlying DHT network as the infrastructure to provide a semantically-based resource discovery would provide a scalable scheme for routing and data location in a large scale network, yet with higher costs of infrastructure maintenance and index management.

In this chapter, it must be noted that not a single work addresses all of the fundamental challenges of the Personalized Web. However, the proposed mechanisms and frameworks in the literature have demonstrated great possibilities to leverage the benefits of P2P overlay networks to address them. The next chapter elaborates on how these P2P networks could be leveraged to enable the Personalized Web.

3.0 PERSONALIZED WEB ARCHITECTURE

The Introduction presented the concept of the Personalized Web as a possible new approach to providing personalized resource discovery, access, and enabling knowledge sharing over the Internet. The Personalized Web differs from other approaches such as search engines in three ways. One of these differences is that the Personalized Web enables users to *discover resources on the Internet based on their individual interests.* It differs by allowing the discovery to *continue to capture the dynamic Internet to reflect the users' interests in a transparent manner.* Finally the Personalized Web enables users to *share and expand their findings with users who have similar interests.*

In order to realize this vision, this thesis proposes an architecture that leverages the benefits from agent technology and P2P overlay technology. The first research question posed is whether it is possible to combine the P2P overlay network and the agent-enabled Semantic Web technologies to provide the foundation for Personalized Web. The previous chapter reviewed the literature of P2P overlay architectures that provide resource discovery, access, and sharing. Additionally, the previous chapter examined the argument for whether or not these architectures could address the fundamental challenges of the Personalized Web and ended with two conclusions. First, the proposed works falled short of addressing all of these challenges. Second, the literature review concluded that a new architecture specifically designed to enable the Personalized Web was needed.

In this chapter, the overlay architecture that addresses these challenges is described. First, an overview of the overlay architecture and its main components is provided to create a foundation. Then the Personalized Web framework defines how these components support Personalized Web functionalities.

3.1 PERSONALIZED WEB ARCHITECTURE

The basic design goal of Personalized Web architecture is to provide users with the ability to adaptively create their own views of the Internet to meet their personal interests in an effective and scalable manner. To achieve this goal, the proposed approach augments the functionalities of a structured overlay network architecture with the capabilities of agent-

based technology in order to provide a flexible and scalable architecture capable of efficiently supporting the main Personalized Web design requirements. The basic architecture is depicted in Figure 16. The architecture comprises three logical layers, namely the *Agent* layer, the *Ontology Overlay Network* (OON) layer, and the *Internet Resource* (IR) layer.

The first two layers rely on the IR to physically locate and access resources. The IR layer uses currently available Internet capabilities to provide these functionalities. The *Agent* layer and the *OON* layer are discussed in more detail in the following sections.

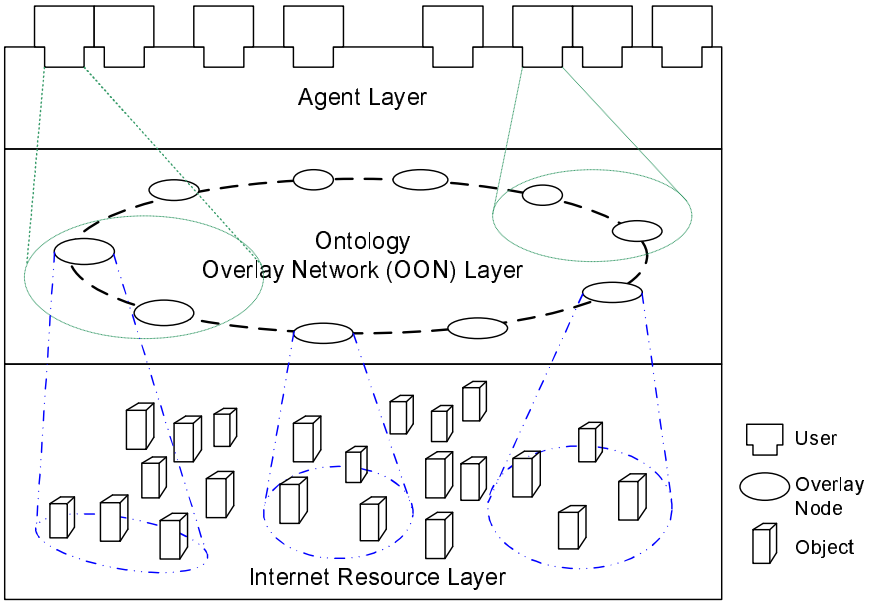

Figure 16: Abstract View of Personalized Web Architecture.

3.1.1 Agent Layer

The main goal of the Personalized Web is to enable a user-defined Internet structure for resource discovery and access. The user's Personalized Web must reflect as closely as possible the user's interests and allow the underlying structure to evolve dynamically in a manner that is transparent to the user. To achieve this goal, the proposed architecture uses an agent-based approach to support the functionalities required. These functionalities allow users

to express personalized views of the Internet, to advertise their resources to communities of similar interests, and to discover new resources of interest based on their personalized similarity profiles.

The basic tenet of the agent layer is a semantically-aware agent, referred to as a "***semlet***," which represents a semantic outlet for a user's resource. The semlet acts as a "surrogate" agent for the associated resource, and, as such, enables this resource to discover and advertise itself to semantically similar resources.

Semlets use the *user-described profile* to extract and dynamically build metadata for the resources. As a result, the metadata reflects the concepts associated with resources of interest to the user. The metadata is then used to advertise the resource to users of communities of interests, and/or locate resources of interest to the user. The capability of a semlet to join other groups of semlets with similar interests provides the basis for the development of a dynamic and distributed graph of semantically related resources that can be retrieved on demand in response to a user's queries.

3.1.2 Ontology Overlay Network (OON) Layer

Personalized Web architecture must allow for the user's personal views of the Internet to evolve dynamically as new resources of interest to the user are created and advertised. Furthermore, this process must occur in a manner transparent to the user, while ensuring that these views continue to reflect the user's interest reliably. To achieve this goal, the proposed architecture uses ontology as a basis for resource representation. Additionally, it employs an ontology overlay network (OON) in order to enable the development of efficient resource indexing and discovery algorithms. In this context, an ontology is defined as a formal, explicit specification of a shared conceptualization, which can refer to the shared understanding of some domains of interests [42].

Underlying the OON architecture is a structured, Distributed Hash Table (DHT) P2P overlay infrastructure that integrates ontology into its structure. Ontology is used to ensure global consistency in semantic classification. As such, given a keyword, the architecture can consistently identify the related concepts. One example of such an ontology can be

derived from Wordnet, a well-known lexical database [31]. The data structure referred to as a Wordnet Ontology Tree (WOT) is presented in *A P2P Overlay Architecture for Personalized Resource Discovery, Access, and Sharing over the Internet* [80]. A WOT is represented by a hierarchy tree, where a node in the tree represents a noun concept described by a set of similar words (synset), and where an edge between two nodes represents the *is-a* relation between these concepts. A partial WOT is shown in Figure 17. Each node in the WOT is assigned a numerical ID that is used as the hash input for a search in ontology overlay network.

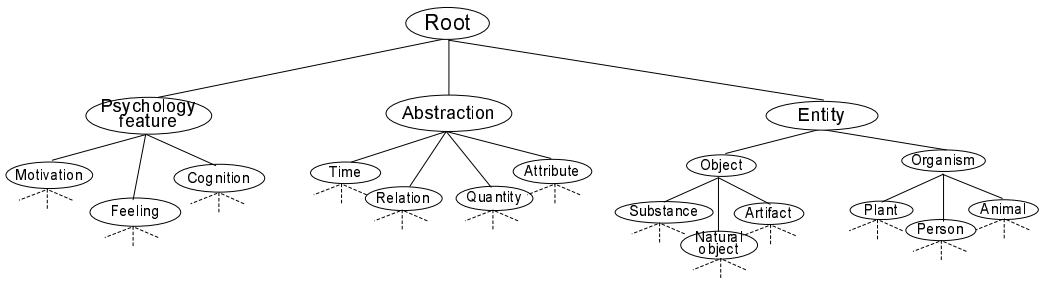

Figure 17: An Example of an Ontology Structure: Wordnet Ontology Tree

The DHT is used to infer a structure that regulates the location and access to a distributed set of resources. Consequently, the OON supports the capability to enable the development of adaptive and scalable search and retrieval algorithms. Using DHT information, overlay nodes, which host resources of interests, can be identified and located deterministically and in a finite number of steps. The identification and location discovery considerably reduce the search overhead due to communication and routing.

In the OON, the resource metadata is clustered, indexed and distributed among overlay nodes based on the semantics described by the associated ontology. Note that only metadata associated with the resource is distributed among nodes in the OON. The actual resource remains physically located at its original site.

3.2 PERSONALIZED WEB FRAMEWORK

In this section, the framework used to support the functionalities of Personalized Web architecture is described. First, the main components of the proposed architecture, as well as their associated data structures and basic functionalities are presented. The mechanisms and schemes used by the overlay architecture for object[1] indexing, advertising and retrieval are then discussed.

3.2.1 Object Metadata

In Personalized Web architecture an object is associated with ***metadata*** which is composed of two main attributes: *"location"* and *"semantics."*

Definition 8. *Metadata of object o, within a community of interest c, can be represented by*

$$m_o = (L_o, S_o^c)$$

where L_o is the location of object o, and S_o^c defines the semantics representing object o, as agreed upon by the members of c.

The *location* attribute is used to locate an object and is typically represented by the object URL. The *semantics* attribute contains the meaning of the associated element as agreed upon by the members of a community of interest. This attribute is represented by a set of concepts defined by the community ontology, referred to as ***Cmm-Ont***. A Cmm-Ont can be viewed as a dictionary of words, how they are related and defined by a community of interest. The Cmm-Ont is used to ensure consistency in semantic classification within the community. Examples of a Cmm-Ont include the WOT presented in the previous chapter and the topic taxonomy used to classify ACM publications.

In practice, a semantic representation of an object can be inferred from keywords that appear in the object's attributes. Typically, an extraction function, referred to as *X()*, scans the object's semantics to extract a set of keywords. The extracted keywords are mapped into the concepts defined by the Cmm-Ont. These concepts are then used to index and advertise

[1] An "object" is used to refer to an instance of a resource in this case.

associated objects within the community of interest. Note that the mapping of keywords into Cmm-Ont concepts is "well-defined" within a community and, as such, is accessible to all members of that community.

3.2.2 Semlets and User Interest Profiles

Objects in a Personalized Web are associated with a semantically-aware agent, referred to as a ***semlet***. A semlet acts as the "*surrogate*" agent to advertise its associated objects and and retrieve similar *objects of interest*, as specified in the user's interest profile.

Definition 9. *Formally, a semlet, associated with a set of objects O, is characterized by a profile, P_{Sl_O}, defined as*

$$P_{Sl_O} = (L_{Sl_O}, PSR_{Sl_O}, SIM_{Sl_O}),$$

where L_{Sl_O} is the semlet locator, PSR_{Sl_O} is the set of personalized semantic rules, and SIM_{Sl_O} is a user-defined similarity metric with respect to objects in O.

The first attribute, L_{Sl_O}, is typically represented by a URL. Notice that a semlet need not reside at the same location as its objects as long as it maintains a pointer to the objects' locations.

The second attribute, PSR_{Sl_O}, is a set semantic rules, referred to as ***Personalized Semantic Rules*** (PSR). These rules are derived from the *user's personal interests* with respect to objects in O. The user's interests can be either explicitly expressed by the user in an interest profile or implicitly derived from objects created by the user. The interests are mapped to a set of PSR using a user-defined mapping function, *U()*. Note that the methods used to express a user's personal interests or derive them from manipulated objects depend on several factors including the type of object and the ontology used by the community. The specification of such methods is outside the scope of this thesis.

For a given object set O and a semlet Sl_O, each rule, included in PSR_{Sl_O}, is a logical expression involving a set of literal concepts and a set of logical operators. These operators typically include *AND*($\wedge$), *OR*($\vee$), and *NEGATE*($\neg$).

Definition 10. *Formally, a set of personalized semantic rules, PSR_{Sl_O} is represented by a tuple*

$$PSR_{Sl_o} = (S_o^u, \otimes),$$

where S_O^u is the concept set associated with user u with respect to objects in O, and $\otimes$ is the set of operators defined on the literal concepts in S_O^u.

Notice that $S_O^u \cap S_O^c \neq \emptyset$, thereby ensuring that the rules used by a user to acquire an object of interest within c are at least partly expressed based on semantics defined by the community c. This constraint is necessary to "integrate" a user's Personalized Web into a specific community of interest without limiting its integration into other communities.

The semlet uses its associated objects' metadata to advertise the objects to members of a community of interest. During its advertising, the semlet registers its interests with respect to objects in O, based on its associated PSR_{Sl_O}. Consequently, when a new object is advertised, the semlet is notified if at least one of the rules in its PSR_{Sl_O} is "*satisfied*" by the semantics of the new object metadata. To illustrate this process, assume that the set PSR_{Sl_O} of semlet Sl_O contains rule $s_i \wedge (s_j \vee s_k)$. A new object advertising metadata containing the concepts $\{s_i, s_k\}$ and $\{s_j, s_k, s_l\}$ causes Sl_O to be notified.

The third attribute, SIM_{Sl_O}, is the user-defined similarity metric used to assess the similarity of the objects, and, as such, determine whether to include the objects in the Personalized Web. This attribute is the key factor that allows the Personalized Web to develop to closely reflect the user's interest. It determines whether retrieved objects are of interest, and how similar these objects are to the user's objects. The metric comprises a similarity function that computes similarity scores between objects and a similarity score threshold. Examples of a similarity function include a Boolean function that counts the number of similar keywords appearing in the objects and an ontology-based similarity function that computes the distance of the path between the concept nodes defined in an ontology graph.

To determine that an object is similar, the semlet, first, computes the similarity score between the user's objects and the object in question using the similarity function. The semlet then compares the similarity score with the similarity threshold. If the similarity score

is within the threshold, the object is considered a similar object; therefore, it is included in the Personalized Web. Otherwise, the semlet drops the object from consideration.

Figure 18 illustrates how a semlet uses associated object metadata, personalized semantic rules, and similarity metrics to develop a Personalized Web. On the left of the figure, the semlet aggregates the objects' semantics needed to perform semantically-based object advertising to the communities of interests. In the middle of the figure, the semlet employs a set of personalized semantic rules to retrieve objects of interest. Coming from the right of the figure, a set of similarity metric(s) is used to find similarity scores between the the retrieved objects and the user's objects. Finally, the retrieved objects, their similarity scores, and the user's objects are used to develop a Personalized Web.

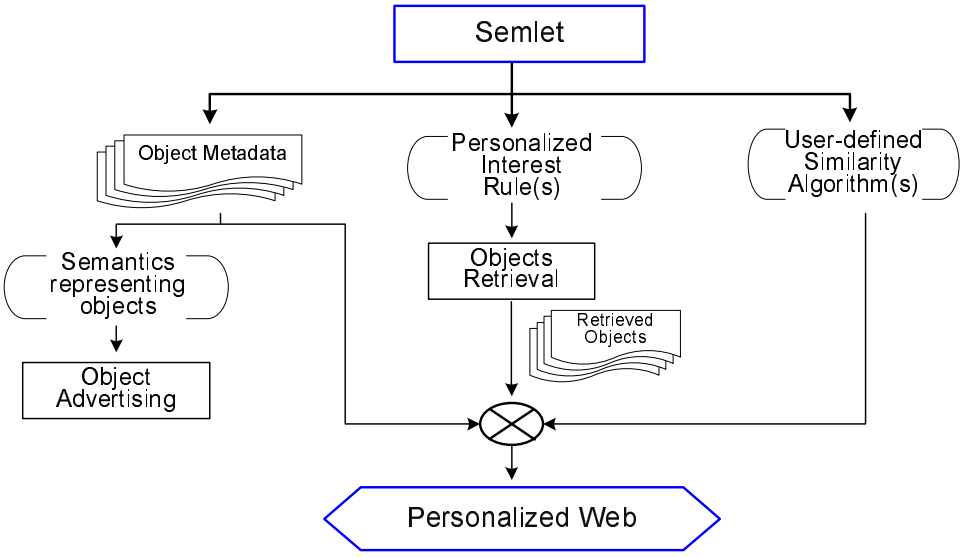

Figure 18: A Semlet-Driven Personalized Web

3.2.3 OON Setup and Organization

In order to support semlet advertising and retrieval in a dynamic, heterogeneous network environment, object metadata and semlet profiles are distributed and indexed among the overlay nodes using a P2P overlay network, referred to as an ***Ontology Overlay Network***

(OON). The OON uses the community's ontology coupled with DHT to infer a *semantic structure* that regulates location and access to a distributed set of objects and semlets. Consequently, the OON provides adaptive and scalable search and discovery to support the functionality of the proposed architecture. To support semantically-based indexing, discovery and advertising, the OON structure makes the following assumptions:

- The Internet is organized into a set, $\mathbf{D}$, of autonomous domains. Each domain, $d \in \mathbf{D}$, elects a node, n_d, to act as an *OON node* in the semantic overlay network. The set of OON nodes, referred to as $\mathbf{N}$, forms the basis of the OON. Hence, $N = \{n_d | d \in \mathbf{D}$ and n_d is an OON node$\}$. An OON is shown in Figure 19.
- In the OON structure, each node, $n_d \in \mathbf{N}$, has a set of neighbors, N_d, which it employs to route data using a DHT-based strategy.
- The nodes in a given domain, d, address the semlet search and advertising to the OON node $n_d \in \mathbf{N}$.
- Let $\mathbf{S}$ represent the set of concepts in Cmm-Ont. A concept $s \in \mathbf{S}$, represents a set of keywords, $K^s = \{k_i^s \in \mathbf{K}, 1 \leq i \leq \| K^s \|\}$, where k_i^s maps to concept s. A keyword, k_i, may refer to multiple Cmm-Ont concepts.
- Let $\mathbf{O}$ be the set of objects in $\mathbf{D}$ and $\mathbf{K}$ be the set of all keywords. Each object $o \in \mathbf{O}$ is represented semantically by a set of concepts $S_o = \{s_i \in \mathbf{S}, 1 \leq i \leq \| S_o \|\}$, which is generated by the object attribute's keyword set $K_o = \{k_i \in \mathbf{K}, 1 \leq i \leq \| K_o \|\}$.
- Each OON node $n_d \in \mathbf{N}$ is assigned a set of semantics S^n, and has the responsibility to advertise and retrieve objects of interest for the semlets associated with any $s_i \in S_d^n$. $\forall s_i \in S^n, hash(s_i) = h_i$ where n_d is responsible for the key h_i. A node that is responsible for s_i is denoted as n^{s_i}.

3.2.4 Personalized Web Components

Many components together support Personalized Web functions. The components are classified into two main groups based on where they are employed. The first group is used at the user node; therefore the components are referred to as *user node components*. The second group of components is used at the OON node; hence, they are referred to as the *OON node*

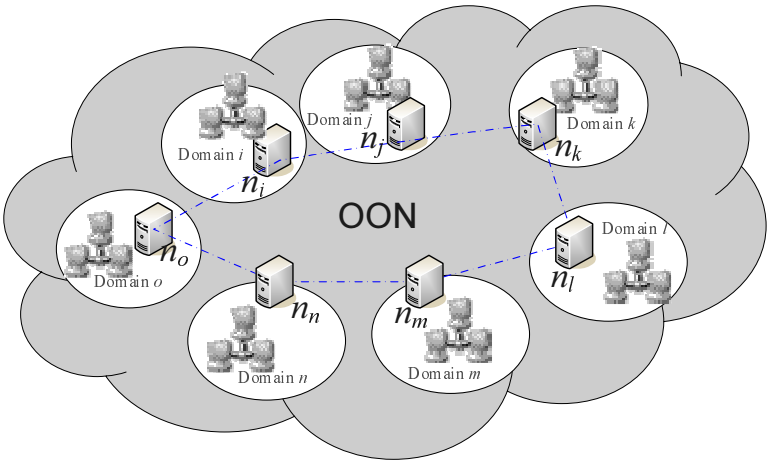

Figure 19: Ontology Overlay Network.

components. The following sections describe these components and their functionalities that support the Personalized Web.

3.2.4.1 User Node Components The user node components are responsible for two main functions: activating semlets to perform object advertising and retrieval as well as managing data storage for the semlets. The group of user node components includes a*semlet handler*, an*object handler*, a *user interest agent*, and a *Personalized Web viewer*. Figure 20 illustrates the user node components and their relations. Their functionalities are as follows:

- The *semlet handler* is responsible for managing the semlets and the storage necessary to hold information related to semlets. Once a user requests the development of a Personalized Web with a set of objects, a set of personalized semantic rules, and a similarity metric, the semlet handler instantiates a semlet, assigns an address to the semlet, and creates the data structures necessary to hold the Personalized Web information, such as objects of interest and communities of interests.
- The *object handler* manages the user's objects and the objects obtained from the semlets. It maintains the objects' locators and the objects themselves allowing the semlets to manage abstract views of these objects.

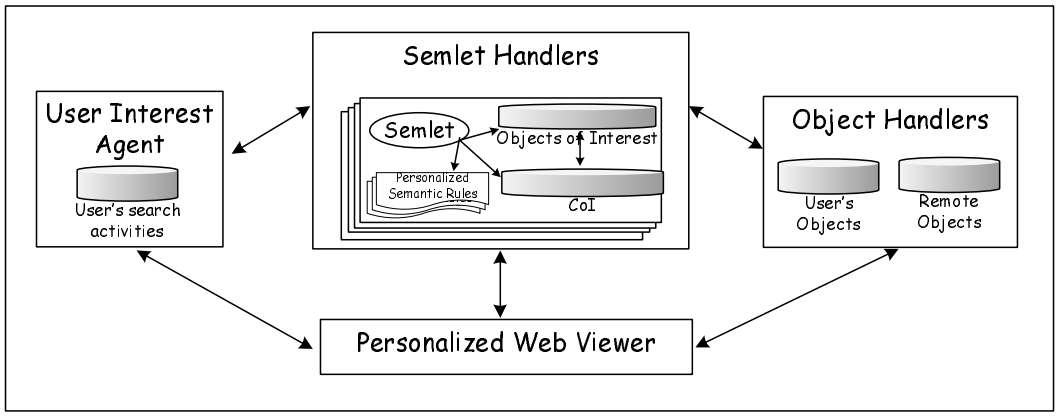

Figure 20: User Node Components.

- The *user interest agent* monitors the user's interest as expressed by the user's search activities. By monitoring the user's search activities periodically, it computes the statistics for the semantics most used during recent activity, and then derives the semantic rules that reflect the user's interest. The interest agent then interacts with semlets to adjust the personalized semantic rules to reflect more accurately the user's interest in a user-transparent manner.
- The *Personalized Web viewer* develops the user's view of a Personalized Web. Once the user requests to view a Personalized Web, the Personalized Web viewer interacts with the associated semlet, acquires objects of interests and their similarity scores, and provides an abstract view of these objects ranked and linked based on their similarity scores.

3.2.4.2 OON Node Components The OON node components are responsible for the performance of four main functions: locating and routing requests to the OON nodes responsible for semlet profile and object metadata, maintaining the advertised object metadata and semlet profiles, finding a match between a semlet profile and object metadata, and delivering the matched object metadata to the semlets. The OON node components include a *DHT component*, a *query handler*, an *object metadata handler*, a *semlet profile handler*, and a *routing handler*. The relations of these components are illustrated in Figure 21 and their functionalities are described below:

- The *DHT component* is responsible for DHT networking functionalities including overlay routing and managing OON node neighbors. It also decides whether the node is responsible for a query. Once an OON node receiving a request, either from a client node or from another OON node, the DHT component checks whether the OON node is responsible for the request. If the node is not responsible for the request, the DHT component forwards the request to the node's neighbor using a DHT-based routing strategy. On the other hand, if the request is for the OON node, the DHT component forwards it to the query handler.

- The *query handler* processes the query. It first extracts the semlet profile and the object metadata from the query. It then interacts with the object metadata handler and the semlet profile handler to search for the object metadata and semlet profiles matching the extracted semantic rules and object metadata. The query handler then marks the ontological concepts contained in the query that have been processed and forwards the query, the matching object metadata and matching profiles to the routing controller.

- The *object metadata handler* is responsible for storing and indexing the advertised object metadata. It indexes the object metadata with the associated ontological concepts for which the OON node is responsible.

- The *semlet profile handler* is responsible for storing and indexing the semlet profile. It also indexes the semlet profile with the ontological concept for which the OON node is responsible.

- The *routing handler* is responsible for routing the query once it is processed by the node. Based on the ontological concepts associated with the query, the routing controller decides weather the query need to be processed further by other OON nodes. If there is at least one ontological concept left to process, the routing controller forwards the query and the matching profile to the DHT component in order to route the next OON node responsible for the concept. If all the ontological concepts associated with the query have been processed, the routing controller then completes the query by returning all the matched object metadata to the requesting semlet and, at the same time, sending the object metadata to all the semlets whose profiles are satisfied by the object metadata.

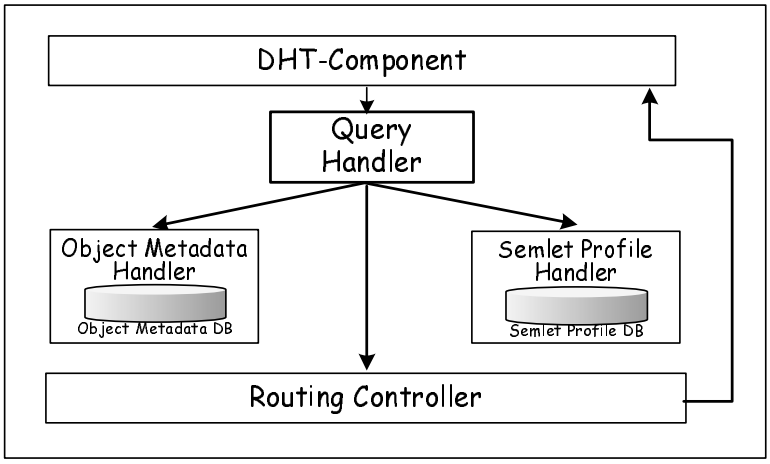

Figure 21: OON Node Components.

To further explain how object metadata and semlet profiles are stored and indexed on an OON node, the Entity Relation (ER) diagram of data on an OON node is illustrated in Figure 22. Beginning with the relationship between an OON node and ontology notes that each OON node supports multiple ontologies. Each ontology contains multiple ontological concepts. However, the OON node is only responsible for some of these concepts. Consequently, the semlet profiles, where personalized semantic rules include at least one of these ontological concepts, are stored and indexed on the OON node. Similarly, the only object metadata that includes one of these concepts in its semantics is stored and indexed on the OON node.

3.3 PERSONALIZED WEB ALGORITHMS AND PROTOCOLS

In this section, the algorithms and protocols to support the main functionalities of a Personalized Web are described. First, the indexing scheme is introduced, followed by a description of the algorithms and protocols that support object advertising and retrieval. Next, the algorithms and protocols necessarily to dynamically form and manage communities of in-

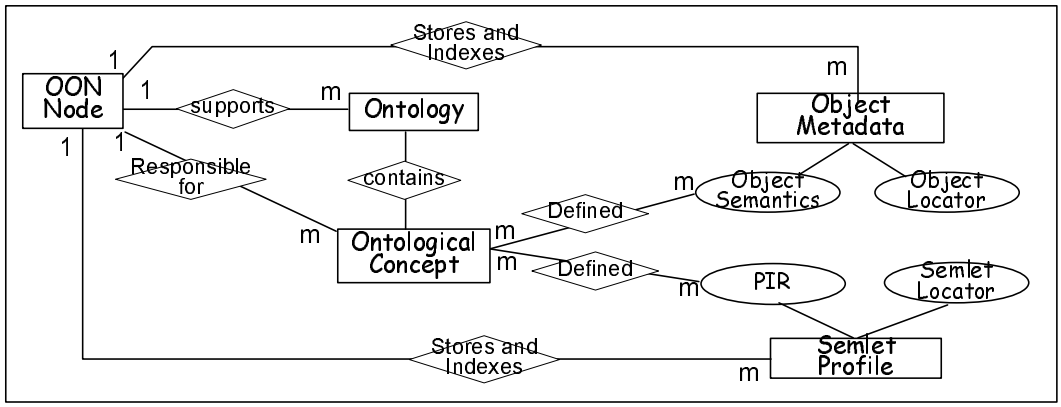

Figure 22: An ER Diagram of Data on an OON Node.

terests are explained. Finally, the algorithms to develop a personal web from the objects acquired by object advertising and retrieval from an OON and communities of interests are presented.

3.3.1 Personalized Web Indexing Scheme

Personalized Web architecture is an agent-driven architecture which achieves its semantic richness through the use of explicit, shared ontologies defined by communities of interests to represent objects. Personalized Web architecture further enhances the DHT-based object distribution scheme by using a unique identifier assigned to each ontological concept as a key to locate the overlay node responsible for maintaining the object and semlet profile indexes associated with the underlying ontology. In other words, the ontology-based hashing scheme utilizes an ontological concept, instead of an object name or a random number, as the hash input to generate the key necessary to distribute the object metadata and semlet profiles on an OON.

Figure 23 depicts the indexing model describing how object metadata and personalized semantic rules are indexed through an OON. Metadata is advertised and indexed in the OON based on its associated ontological concepts. These ontological concepts refer to object semantics extracted from the object attributes. The concepts are hashed using a system-

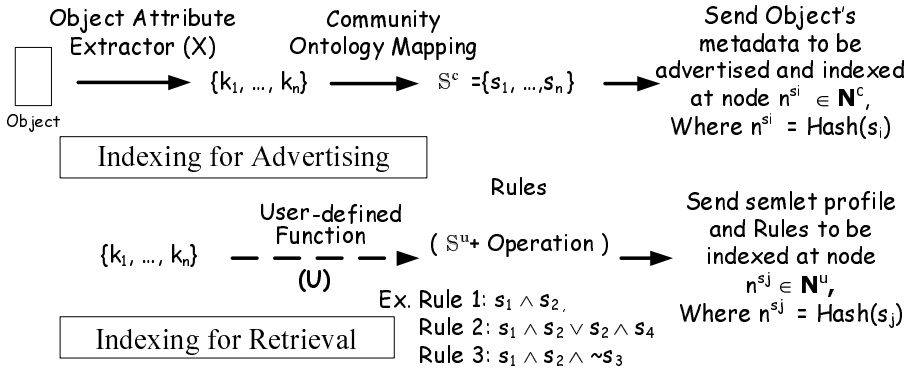

Figure 23: Personalized Web Indexing Model.

wide hash function to obtain the routing keys. These keys are then mapped to the OON nodes. As a result, these OON nodes become responsible for storing and indexing the object metadata. A semlet profile is stored and indexed in the OON using the same mapping scheme. However, in the case of semlet profiles, the mapped concepts are those extracted from the personalized semantic rules.

3.3.2 Semlet-Driven Advertising and Retrieval

Semlet-driven advertising and retrieval is the key to acquiring objects of interest so as to develop a Personalized Web in a user-transparent manner. It utilizes the autonomy of the semlet agent and the persistence of the OON's data location to allow users to automatically exchange objects of interests in an efficient manner.

The strategy underlying semlet advertising and retrieval is that the semlet advertises "*aggressively*" and retrieves "*selectively.*" That is, each semlet advertises its objects to ***all*** other semlets whichever has at least one personalized semantic rule that matches the object semantics. These semlets are referred to as potential interest semlets. On the other hand, the semlet retrieves ***selectively*** only objects that are similar to its associated objects in the development of its Personalized Web. In fact, an object that has metadata matching the semlet profile may be retrieved. However, this object will only be considered as an object

of interest when its similarity score with respect to the user's objects falls within a specified range. As a result, the Personalized Web contains only objects of interests that are similar to the user's objects; therefore, it closely reflects the user's interests.

In order to perform semlet advertising and retrieval in a scalable manner, the OON acts as the rendezvous point of discovery as well as the proxy of object advertising and retrieval for semlets. The OON stores and indexes the semlet profiles and object metadata based on their associated semantics. The similar object metadata and semlet profiles are stored and indexed on the same set of OON nodes. Upon receiving queries, OON nodes consistently route the queries towards the nodes responsible for them using a DHT-based routing strategy. As a result, the queries are routed towards a set of OON nodes that contain semantically related semlet profiles or objects metadata. Once the queries arrive at the responsible OON nodes, the OON nodes search for a match, and perform advertising and retrieval for the semlets. For an advertising query, the OON node sends the object metadata to the semlet that has a matching profile. Similarly, for a retrieval query, the OON node sends the matching object metadata back to the semlet. The step-by-step procedure is illustrated in Figure 24 to further explain semlet advertising and retrieval, and is described below:

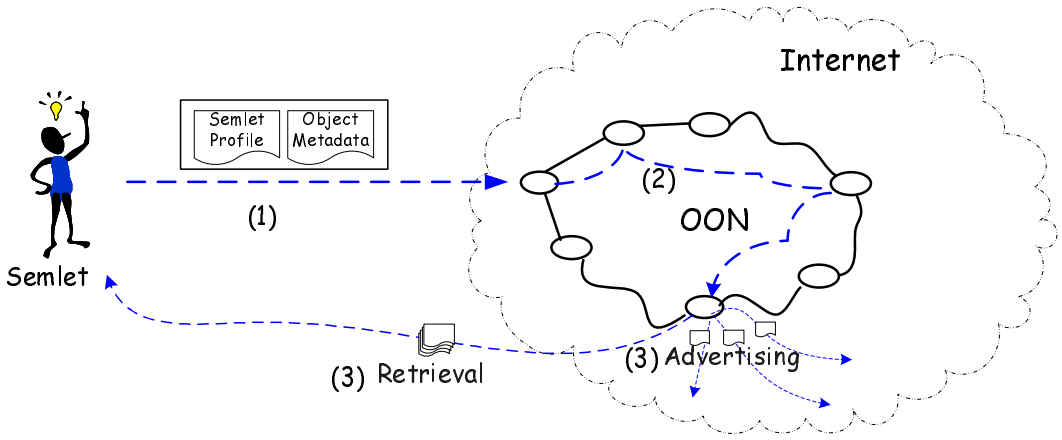

Figure 24: Personalized Web Object Advertising and Retrieval.

1. The semlet first locates an OON node within its own domain.
2. The semlet then sends a query which contains the object metadata and its profile to this OON node (step (1) in the figure). Note that once sending the semlet profile to the

OON, the semlet makes itself known as "active" to the OON. The semlet then remains passive on the client node, waiting for the query result and an advertisement of a newly added object that matches its profile.

3. The OON node then routes the query to the OON node responsible for one of the hash keys of its associated ontological concepts (step (2) in the figure).
4. Upon the arrival of the query, the OON node responsible for the key executes the following steps:
   a. First, the OON node discovers semlet profiles whose rules are satisfied by the metadata carried in the query. The OON node also finds objects whose metadata are satisfied by the rules in the query.
   b. The OON node may store or index the object metadata and semlet profile contained in the query for later discovery and notification.
   c. The OON node then forwards the matching profiles and object metadata to the next OON node responsible for the next concept in the associated concept set.
5. The forwarding continues until all OON nodes responsible for the concepts in the concept set are visited.
6. At the last OON node, the OON node notifies all semlets associated with the collected profiles about the object, and sends the collected object metadata back to the advertising semlet (step (3) in the figure).

Note that the strategy described above uses *recursive* routing which assumes each OON node responsible for the key computes the next key for the query and forwards the query and the discovery result (the matched profiles and object metadata) to the OON node responsible for the next key. However, an alternative approach referred to as *iterative* routing can be employed. Using the iterative routing, the semlet computes all the keys, composes all the queries, and submits all the queries separately to the OON nodes responsible for the keys. As a result, the discovery results are returned separately from each of the OON nodes.

These two approaches can be differentiated because the recursive approach distributes executions among OON nodes and the iterative approach centralizes execution using a semlet. The recursive approach uses bandwidth more efficiently than the iterative approach does. However, the recursive approach is more prone to incomplete routing due to the failure of

OON nodes. A discussion on the advantages and the pitfalls of these two approaches can be found elsewhere [22].

The scalability and robustness of Personalized Web architecture depend largely on how the OON performs object advertising and retrieval for the semlets. Thus, three object advertising and retrieval schemes which further explore the key factors of the scalability and robustness of a Personalized Web are presented in Chapter four.

3.3.3 Formation and Management of Communities of Interests (CoI)

One of the main goals of Personalized Web architecture is to allow users to find users who have a similar interest in order to form communities of interests (CoI) in a user-transparent manner. CoIs help users to share objects of interests in an efficient manner due to the fact that objects belonging to a member of a CoI will likely interest other members in that CoI. By advertising and discovering objects of interests among members of a CoI, semlets effectively acquire objects of interest among them to develop their Personalized Webs.

Personalized Web architecture allows semlets to form CoIs and manage members of their CoI while performing object advertising and retrieval. During object advertising and retrieval, a semlet discovers objects of interest and the semlets that own these objects. These semlets are used as a basis to form a CoI. The advertising semlet can form its own CoI with semlets that have similar objects as the members of the CoI. At the same time, the semlets who are interested in the advertised objects can add the advertising semlet to their CoIs. Consequently, CoIs are dynamically formed and adaptively adjusted to reflect the current interests expressed via object advertising and retrieval.

Due to the fact that each semlet uses its own personalized semantic rules and user-defined similarity metrics, semlets independently develop their CoIs. Each semlet applies a user-defined similarity metric to compute similarity scores which are used to determine members of the CoI. As a result, two semlets who are interested in each other's objects may have different CoI members if their associated similarity metrics are different. The user-defined similarity metric verifies two main properties, namely *asymmetry* and *non-transitivity*.

- *Asymmetry*: Interests between semlets are asymetric. For example, when semlet A is interested in the objects associated with semlet B, it does not imply that semlet B has an interest in the objects associated with semlet A.
- *Non-transitivity*: Interests among semlets are not transitive. For instance, the interest of semlet A in object B and the interest of semlet B in object C does not necessarily imply that semlet A has interest in object C.

Mindful of the above properties, semlets independently develop their CoIs. For example, semlet A and B have object o_A and o_B. Semlet A is interested in object o_B, when $Sim^A(o_A, o_B) \leq \delta_A$, where $Sim^A(*)$ is the similarity function for semlet A and δ_A is a similarity threshold for semlet A. On the other hand, semlet B is interested in object o_A, only if $Sim^B(r_B, r_A) \leq \delta_B$. The similarity threshold (δ) is used to trigger advertising to and retrieval from other semlets.

To efficiently manage CoIs, each semlet forms an *Advertising CoI*(A-CoI) for the members who have an interest in its objects, and a *Retrieval CoI*(R-CoI) for the members who have objects that interest the semlet. A-CoI is referred to as an effective advertising group. Whenever a semlet advertises an object to the group, the members of the group will have a high probability of finding a match to the members' interest. On the other hand, since the semlet is likely to find objects of interests from the members of this group, R-CoI is employed as an effective retrieval group.

Having both CoIs, a semlet effectively and efficiently performs object advertising and retrieval. The semlet does not need to go through the OON to discover objects of interest. Instead, the semlet can directly use R-CoI to retrieve objects of interests and A-CoI to advertise its newly added objects. The semlet only employs the OON when it needs to acquire new members of its CoIs or to make itself known to the other semlets.

One may also use the CoIs to improve the object sharing among members. For instance, semlets can give their advertising priority to the semlets that are in its R-CoI to motivate users to share their objects. Using such a policy, semlets may advertise their new objects only to the semlets that have objects of interest to them. The semlets that want the objects advertised to them must share their objects aggressively. As a result, object sharing is improved among CoI members.

3.3.4 Personalized Web Development

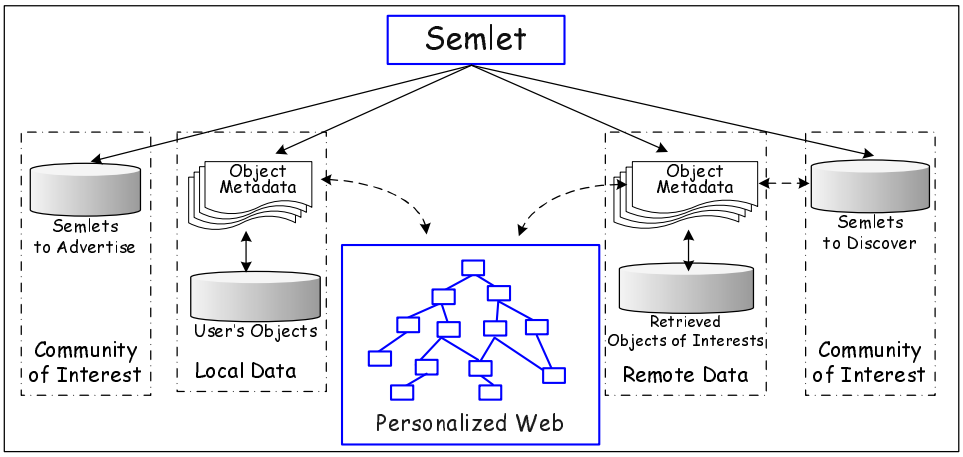

Figure 25: User Node Components.

A Personalized Web is developed from objects that reflect the user's interest. Contained in the Personalized Web, the user's objects and others objects form a graph of objects of interest. The central part of the graph contains the user's objects which are linked to other users' objects. These objects are considered as similar based on their similarity profiles.

The Personalized Web evolves through time as semlets perform object advertising and retrieval. The semlet retrieves by itself or receives from other semlets objects of interest through the OON and its CoI members. As the objects of interests are added to or removed from a Personalized Web, the graph structure is modified to reflect the similarity of the objects of interest. As a result, the Personalized Web is efficiently developed in a user-transparent manner, and it effectively reflects the user interests and the availability of the objects of interest on the Internet. In other words, the Personalized Web provides an updated view of the objects specific to the user's interest at any convenient time. Figure 25 illustrates a semlet and its related components including A-CoI, R-CoI, local data, object data, and how these components interact with the semlet to develop a Personalized Web.

3.4 CONCLUSION

This chapter described a Personalized Web architecture that leverages agent technology and P2P overlay network technology to enable personalized object discovery and access Internet objects to reflect users' interests. Personalized Web components including a semantically-aware agent referred to as a "semlet" and a DHT P2P overlay infrastructure referred to as an "OON," were presented. The description of this architecture and these components fulfills the main objective of describing how semlets and OONs were used to develop Personalized Webs for users in an efficient and scalable manner.

Semlets were used to act on behalf of users to automatically advertise and retrieve objects of interests, dynamically form and manage communities of interests (CoI), and persistently develop a personal web[2] of the objects of interest to the users. The OON, on the other hand, was employed as the rendezvous point for semlets and as the proxy for object advertising and retrieval in order to allow semlets to perform their functionalities in an efficient and scalable manner. The OON helps semlets to efficiently find and share objects of interests. At the same time, the OON enables semlets to discover semlets who share similar objects to form CoIs. In turn, the CoIs allow semlets to efficiently advertise and retrieve objects of interests among them.

The key mechanism to developing a Personalized Web is object advertising and retrieval based on an ontology agreed upon by a CoI referred to as a *Comm-Ont*. A Comm-Ont is used to ensure that the semantics describing objects and the semantics expressing the users' interests are consistent with one another within the CoI. Ontological concepts defined by a Comm-Ont are employed to represent the semantics of objects advertised in the CoI. The ontological concepts are also used to represent *personalized semantic rules* used to retrieve objects of interest.

To develop a Personalized Web, a semlet is assigned a set of objects and a specific interest expressed in terms of *personalized semantic rules* and *similarity metrics*. The semlet then sends an advertising and retrieval query that includes the object metadata and its semlet profile to the OON. The object metadata contains the locations of the objects and the

[2]A personal web is a product of the actualization of the Personalized Web concept.

ontological concepts representing the object semantics. It is used to advertise the user's objects to CoI. On the other hand, the semlet profile contains the semlet's address and the personalized semantic rules. It is used to retrieve the objects of interest for the user. The OON then routes the query and performs object advertising and retrieval for the semlet.

Aimed at supporting the semlet functionalities, the OON provides indexing and routing infrastructure for the semlets. Each OON node is assigned a random set of ontological concepts defined by the Comm-Ont using a DHT-based object mapping scheme. Each node is also responsible for routing the queries to the node responsible for them. Upon receiving a query, an OON node routes the query to other nodes or, if the node is responsible for one of its associated concepts, processes it. The OON node processes the query by first finding matching object metadata and matching the semlet profiles. Then it stores and indexes the associated object metadata and semlet profile on the node. Finally it either forwards all the the matching profiles to the next responsible node or, if it is the last node responsible for the query, performs object advertising or retrieval for the semlet. As semlets continue to advertise new objects to the OON, the semlets, that are interested in the objects and previously sent their profiles to the OON, are notified. Consequently, their associated Personalized Webs continue to evolve in an efficient manner.

In conclusion, the chapter has described the main components, data structures, mechanisms, and protocols necessary to realize the Personalized Web. The scalability and robustness of the Personalized Web depends mostly on how the semlets perform object advertising and retrieval using the OONs. For this reason, the next chapter focuses on object and advertising schemes to improve robustness and scalability of Personalized Web.

4.0 SEMLET ADVERTISING AND RETRIEVAL SCHEMES

This chapter explores the mechanisms and protocols needed to support the OONs so that they can perform object advertising and retrieval in an effective and scalable manner. Chapter three discusses the idea that object advertising and retrieval is the key mechanism necessary to develop a Personalized Web, and the idea that OONs are used to perform object advertising and retrieval. Consequently, to improve the scalability and robustness of the Personalized Web, the OONs must efficiently manage storage and communications used for advertising and retrieval.

To explore scalable and robust mechanisms and protocols that support the OONs, three advertising and retrieval schemes will be presented, namely the *Aggressive* scheme, the *Crawler-based* scheme and the *Min-Cover-Rule* scheme. These schemes are designed with different performance objectives, which result in different benefits and drawbacks in various circumstances. Such circumstances are also explored in this chapter. First, the basic models of the Personalized Web architecture's main components as well as a case study of two semlets performing object advertising and retrieval are introduced. Next, each scheme is described, followed by a discussion of its benefits and drawbacks. Finally, the chapter concludes with a discussion of the various circumstances where each scheme is beneficial.

4.1 BASIC MODELS FOR SEMLET ADVERTISING AND RETRIEVAL

This section commences with the summarization of Personalized Web models, including a description of all the components and how these components are used to find objects of interests. Such models are used to explain the advertising and retrieval schemes in the sections that follow. Additionally, a scenario of two semlets performing object advertising and retrieval, discussed below, demonstrates how each scheme works and how one scheme differs from the others.

4.1.1 Personalized Web Component Models

A brief summary of Personalized Web models is helpful before going on to discuss the different advertising and retrieval schemes. Three main components necessary to develop a Personalized Web are a semlet agent that automates the object advertising and retrieval, a set of user objects associated with the web, and the user-defined semantic rules to retrieve objects to the web. This section briefly describes and formally defines these three components. The discussion moves on to explore an example of one satisfaction criterion used to determine whether an object semantic described by an object metadata satisfies a personalized semantic rule. The satisfaction criterion used on the OON node determines whether the advertised objects should be retrieved from or advertised to semlets.

The first component of a Personalized Web is a semlet. Each semlet is associated with a set of objects, a set of personalized semantic rules and a similarity metric to carry out two main jobs. First, it advertises the associated objects to the CoI based on semantics defined by a community ontology. The community ontology ensures that the semantics used in advertising and retrieval are consistent among the members of the community. Second, the semlet retrieves objects of interest for the user. It uses the personalized semantic rules to collect objects of interest, and then employs a user-defined similarity metric to filter and rank these objects based on their similarities to the user's objects.

- Formally, a semlet, associated with a set of objects O, is characterized by a profile, P_{Sl_O}, defined as

  $$P_{Sl_O} = (L_{Sl_O}, PSR_{Sl_O}, SIM_{Sl_O}),$$

  where L_{Sl_O} is the semlet locator, PSR_{Sl_O} is the set of personalized semantic rules, and SIM_{Sl_O} is a user-defined similarity metric with respect to objects in O.

Each object is represented by an object metadata which comprises an object locator and a set of ontological concepts defined by a community ontology. The ontological concepts are used to represent the object semantic. These concepts are also referred to as the Boolean literals of the ontological concepts that hold a "*true*" value. Consequently, an object with semantic $\{s_1, s_2, s_3\}$ refers to the object that are described by the *true* literals of s_1, s_2, and s_3.

- Metadata of object i, within a CoI c, can be represented by

$$m_i = (L_i, S_i^c)$$

where L_i is the location of object i, and S_i^c defines the semantic representing object i, as agreed upon by the members of c.

A personalized semantic rule is composed of a set of ontological concepts defined by a community ontology and a set of binary Boolean operators. A semantic rule can be viewed as a Boolean expression, where the literals of the expression are represented by ontological concepts, and the operators of these literals are represented by the binary Boolean operators.

- Formally, a set of personalized semantic rules, PSR_{Sl_O} is represented by a tuple

$$PSR_{Sl_O} = (S_O^u, \otimes),$$

where S_O^u is a set of ontological concepts associated with user u with respect to objects in O, and $\otimes$ is the set of binary Boolean operators defined on the literals of the concepts in S_O^u.

An object metadata is considered of interest when its semantics "satisfy" one of the personalized semantic rules. That is, an object is considered of interest when its semantics hold the ontological literals that result in at least one of the semantic rules receving a *true* result. For instance, an object semantic $\{s_1, s_2, s_3, s_4\}$ satisfies the rule $\{s_1 \wedge s_2 \wedge s_4\}$, and rule $\{s_1 \wedge s_5 \vee s_2 \wedge s_3\}$, because all of the ontological concepts s_1, s_2, s_3, s_4 are *true*. This leads $\{s_1 \wedge s_2 \wedge s_4\}$ and $\{s_2 \wedge s_3\}$ to be *true*. On the other hand, the object semantic does not satisfy rule $\{s_1 \wedge s_4 \wedge s_5\}$, and rule $\{s_1 \wedge s_5 \vee s_4 \wedge s_5\}$.

- Formally, PSR_{Sl_O} is satisfied by an object metadata v if $PSR_{Sl_O}(T(S_v^c)) = true$, when $T(S_v^c)$ refers to the set of true literals associated with the ontological concept set S_v^c of the object v and $PSR_{Sl_O}(T)$ is the OR-result of all rule evaluations on the truth values set T.

4.1.2 Semlet Query

A semlet query is used to find objects and semlets of interests in an OON. Once a semlet performs object advertising and retrieval, the semlet sends a semlet query to one of the OON nodes. The semlet query then propagates through the OON to collect semlet profiles and metadata of objects of interests.

Each semlet query carries five basic components: a set of routing keys needed to propagate to the target nodes in the OON, a set of object metadata to be advertised, a semlet profile to retrieve objects of interest, a set of collected object metadata, and a set of collected semlet profiles. The routing keys are used to route the query to the OON nodes that are responsible for the related object metadata indexes. The object metadata is used to find semlets whose semantic rules satisfy the the object metadata. The semlet profile is employed to discover objects whose metadata satisfies a semantic rule associated with the profile. The last two components are used to hold the object metadata and the semlet profiles of objects of interests found on the OON nodes. Figure 26 depicts the query components.

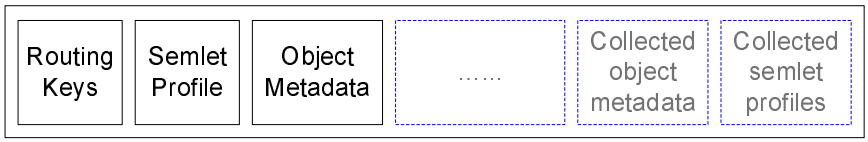

Figure 26: A Semlet Query.

Note that the discovered object of interest's metadata and semlet profiles are forwarded with the query to the last OON node responsible for the last routing key. This node then performs object notification. The forwarding of the discovered objects and semlet profiles of interest helps to avoid multiple notifications to the semlets.

4.1.3 A Scenario of Semlet Advertising and Retrieval

To further explain how the query propagates and performs advertising and retrieval, a scenario of two semlets submitting queries to the OON is described below. The two semlets, Sl_o and Sl_v, are defined as follows:

- The semlet Sl_o is associated with object o, with the object semantic $S_o^c = \{s_1, s_2, s_3\}$, and the semantic rule $r_o = \{s_2 \wedge (s_1 \vee s_3)\}$.

- The semlet Sl_v is associated with object v, with the object semantic $S_v^c = \{s_2, s_3, s_5\}$, and the semantic rule $r_v = \{s_2, s_3, s_5\}$.

There also exists an OON composed of five nodes, $n^{s_1}, n^{s_2}, n^{s_3}, n^{s_4}, n^{s_5}$, where node n^{s_i} is assumed to be responsible for ontological concept s_i. Figure 27 depicts this scenario.

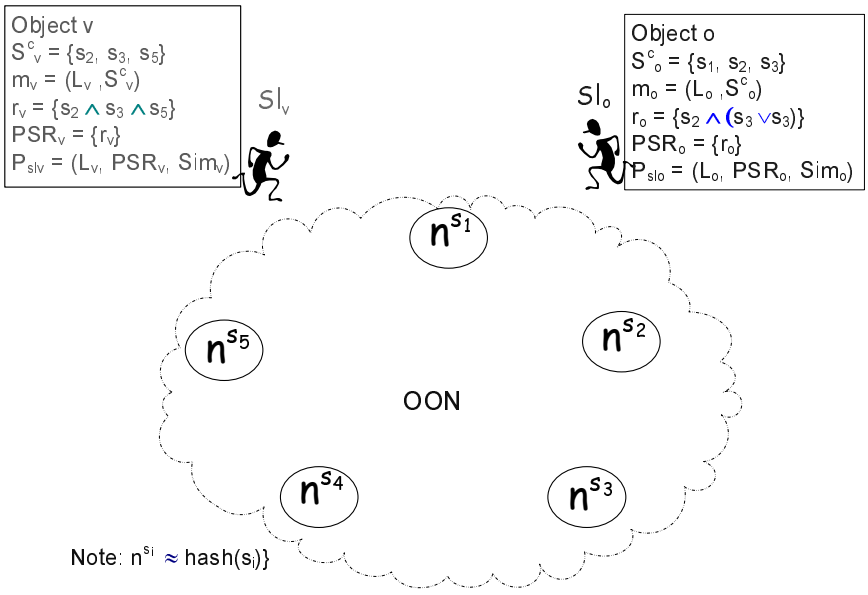

Figure 27: Example of Semlet Advertising and Retrieval.

From such a scenario, where the information is known, one can conclude that Sl_o is interested in object v since $R_o(T(S_v^c)) \rightarrow true$, and Sl_v is not interested in object o. However, in the Internet environment where the other semlet information is unknown, mechanisms and protocols are needed to allow a query to propagate to the right information sources. This chapter focuses on the exploration of such effective and efficient mechanisms and protocols through three alternative object advertising and retrieval schemes, namely the aggressive scheme, the crawler-based scheme, and the min-cover-rule scheme. These schemes are discussed in the sections that follow.

4.2 AGGRESSIVE SCHEME

The "*aggressive*" scheme uses storage and communication aggressively to support semlet advertising and retrieval. Using this scheme, all the OON nodes responsible for the query perform data indexing and storing. The OON nodes store and index the query's object metadata and semlet profiles for later discovery and advertising.

The pseudo-code demonstrating advertising and retrieval using this scheme is described below. The code is divided into the functions performed by a semlet and by an OON node. The first function used by a semlet to start object advertising and retrieval. The second function exploited by a semlet when it receives an advertised object metadata. The OON performs the third function when the node receives a semlet query. Note that the second function is used by all schemes. It is described here for completeness, not for comparison purposes.

Semlet: `Starts object advertising and retrieval (given an object metadata` $m_o = (L_o, S_o^c)$`, and a semlet profile` $P_{Sl_o} = (SL_o, PSR_o, SIM_o)$`, where` $PSR_o = \{r_o\}$`, and` $r_o = (S_o^u, \otimes)$`)`

```
1: //obtain routing keys RK_o
2: S_o ← S_o^c ∪ S_o^u
3: for all s ∈ S_o do
4:   adds hash(s) to RK_o
5: end for
6: selects one key k_i from RK_o
7: composes a query query_o ← [k_i, RK_o, m_o, P_Sl_o]
8: sends the query to an OON node located within its domain
9: waits for the returned query or advertised object metadata from other semlets
```

Semlet: `Receives an advertised object metadata` $m_v = (L_v, S_v^c)$`,`

```
1: retrieves object v from L_v
2: computes a similarity score Sim_O(v, O)
3: if the similarity score > Threshold_O then
4:   adds v to O
5:   adjusts object graph to reflect the similarity among objects in O
6:   adds Sl_v to the retrieval community of semlet Sl_O, R-CoI_O
7:   notifies Sl_v to add Sl_o to its advertising community, A-CoI_V
8: end if
```

OON: Receives a $query_o = [k_i, RK_o, m_o, P_{Sl_o}]$

```
1: if the OON node is not responsible for k_i then
2:   forwards the query to the neighbor that makes the most progress towards the
     node responsible for k_i
3: else
4:   //collect semlet profiles to advertise
5:   if the OON node is responsible for any concepts in S_o^c then
6:     retrieves the semlet profiles satisfied by m_o
7:     stores and indexes m_o on the node
8:   end if
9:   //collect metadata of objects of interest
10:  if the OON is responsible for any concepts in S_o^u then
11:    retrieves the object metadata that satisfies the semantic rule r_o
12:    stores and indexes P_{Sl_o} on the node
13:  end if
14:  removes the keys that this node is responsible for from RK_o
15:  if  no routing keys left in RK_o  then
16:    advertises m_o to the semlets associated with the collected semlet profiles

17:    returns all the collected object metadata to the querying semlet
18:  else
19:    selects a key k_j from RK_o
20:    forwards the query, the collected object metadata, and semlet profiles to
       the node's neighbor that makes the most progress towards the OON node
       responsible for the k_j
21:  end if
22: end if
```

Figure 28 depicts the scenario to further explain how the query propagates and performs advertising and retrieval using the aggressive scheme. In this example, a semlet Sl_o first arrives at the OON. Based on the semantics of interest $\{s_1, s_2, s_3\}$, it sends a query to propagate to nodes n^{s_1}, n^{s_2}, and n^{s_3} . Once the query arrives n^{s_1}, the OON node stores and indexes the object metadata and the profile. It then forwards the query to the next node n^{s_2}. At n^{s_2} and n^{s_3}, the OON nodes do the same. The query then stops at n^{s_3} since there are no more nodes to forward to, and no objects of interest are found.

Later, the semlet Sl_v arrives at the OON. It then sends the query to propagate on n^{s_2}, n^{s_3}, and n^{s_5}. At n^{s_2} and n^{s_3}, the OON node finds the rule r_o of Sl_o satisfied by m_v, it then forwards profile P_{Sl_o} to the next node. At the last propagation node n^{s_5}, the OON node notifies m_v to Sl_o.

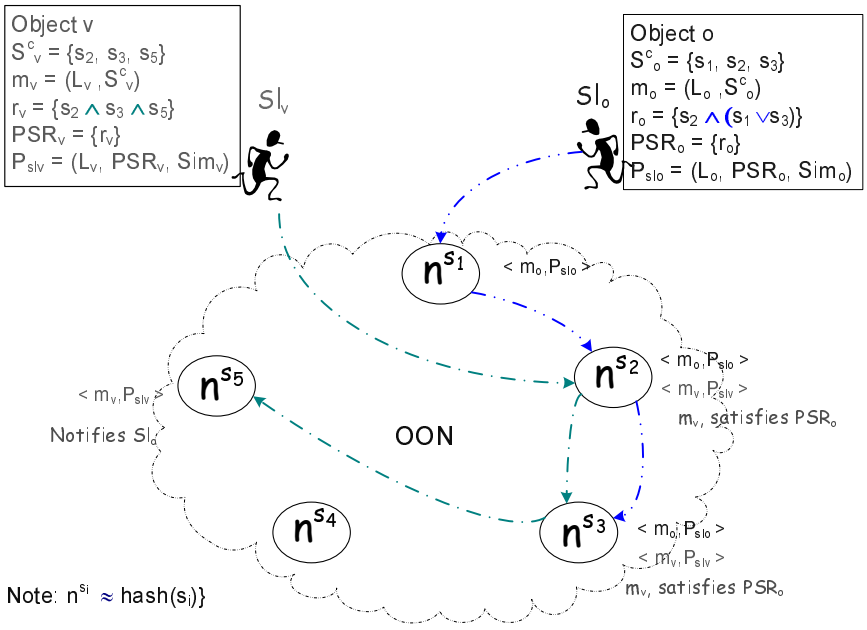

Figure 28: Aggressive Scheme.

4.2.1 Costs and Benefits

The advantage of the aggressive scheme is its added ability to deal with node failures. This ability is achieved through object metadata replication. One drawback of the aggressive scheme is the need to use multiple overlay nodes to store semlet profiles and object metadata. This activity may lead to excessive overhead in terms of storage. To address this shortcoming, a crawler-based scheme, which trades storage at the cost of increasing communication overhead, is described next.

4.3 CRAWLER-BASED SCHEME

Similar to the aggressive scheme, the crawler-based scheme advertises and retrieves objects of interest by sending a query to all the nodes that are responsible for its associated concepts. However, the object metadata and semlet profile will be stored only in a subset of these nodes.

Using this strategy, a semlet may miss an object notification. Consider the case that occurs when the metadata of an object v with concepts $\{s_i, s_j, s_l\}$ is stored at n^{s_l}. Later, a semlet Sl_o arrives with interest rules $s_i \wedge s_j$ and decides to store its profile at node n^{s_i}. It then visits n^{s_i}, and n^{s_j}. Consequently, the semlet misses the object v.

To resolve this problem, semlets no longer wait passively for notifications. Instead, a semlet periodically sends refresh messages to update previously advertised objects and potentially harvests new objects of interest from the OON node. In the case above, when the semlet of object v does the refreshing, it discovers the newly added semlet profiles, and advertises to those semlets that missed advertising previously. The advertising and retrieval mechanism of this scheme is similar to the aggressive scheme, except that the subset of keys representing metadata and rules are stored for later notification and advertising. In addition, each semlet schedules an alarm to refresh a query periodically. The OON nodes must also maintain time-stamps for the object metadata and the semlet profiles stored on their nodes. Such that, when a refreshing query arrives, the OON nodes only advertise and retrieve the new objects of interest. To further describe how this scheme supports semlet advertising and retrieval, the main algorithms are described below. Note that the lines of code that differ from those of the aggressive scheme are presented in bold font.

Semlet: `Starts object advertising and retrieval (given an object metadata` $m_o = (L_o, S_o^c)$`, a semlet profile` $P_{Sl_o} = (SL_o, PSR_o, SIM_o)$`, where` $PSR_o = \{r_o\}$`,` $r_o = (S_o^u, \otimes)$`, and an update interval` $u_{interval}$`)`

`1: //obtain routing keys` RK_o
`2:` $S_o \leftarrow S_o^c \cup S_o^u$
`3: for all` $s \in S_o$ `do`
`4:  adds` $hash(s)$ `to` RK_o
`5: end for`
6: picks a subset of ontological concepts representing the object metadata
7: computes keys for the concepts, and adds them to MK_o
8: picks a subset of ontological concepts associated with each semantic rule
9: computes keys for the concepts for all the rules and adds them to PK_o
`10: selects one key` k_i `from` RK_o
`11: composes a query` $query_o \leftarrow [k_i, RK_o, m_o, P_{Sl_o}, MK_o, PK_o, u_{interval}]$
`12: sends the query to an OON node located within its domain`
13: sets up an alarm to re-submit the query
`14: waits for the returned query or an advertised object metadata from other semlets`

OON: Receives $query_o = [k_i, RK_o, m_o, P_{Sl_o}, mK_o, PK_o, u_{interval}]$

```
1: if the OON node is not responsible for k_i then
2:   forwards the query to the neighbor that makes the most progress towards the
     node responsible for k_i
3: else
4:   //collect semlet profiles to advertise
5:   if the OON node is responsible for any concepts in S_o^c then
6:     retrieves profiles that are stored after the last query time (the
       current_time − u_interval ), and that contain the associated rules satisfied by
       the object metadata m_o
7:     if the OON node is responsible for any k ∈ MK_o then
8:       if m_o is not stored on the node then
9:         stores and indexes m_o with the current time as the time_stamp on the node
10:      end if
11:    end if
12:  end if
13:  //collect metadata of objects of interest
14:  if the OON is responsible for any concepts in S_o^u then
15:    retrieves object metadata that is stored after the last query time and that
       satisfies the semantic rule r_o
16:    if the OON node is responsible for any k ∈ PK_O then
17:      if P − Sl_o is not stored on the node then
18:        stores and indexes P_Sl_o with the current time as the time_stamp on the
           node
19:      end if
20:    end if
21:  end if
22:  removes the keys that this node is responsible for from RK_o
23:  if  no routing keys left in RK_o  then
24:    advertises m_o to the semlets of interest
25:    return all the collected object metadata to the querying semlet
26:  else
27:    selects a key k_j from RK_o
28:    forwards the query, the collected object metadata, and semlet profiles to
       the node's neighbor that makes the most progress towards the OON node
       responsible for the k_j
29:  end if
30: end if
```

Figure 29 illustrates using this scheme for the scenario described in Section 4.1.3. A semlet Sl_o associated with object o first arrives at the OON. Based on the semantics of interest $\{s_1, s_2, s_3\}$, it sends a query to propagate to nodes n^{s_1}, n^{s_2}, and n^{s_3}. However, the OON node only stores and indexes the object metadata and the profile on one of these nodes, assuming n^{s_2}.

Later, the semlet Sl_v associated with object v arrives at the OON. The semlet then sends the query to propagate to n^{s_2}, n^{s_3}, and n^{s_5}. At n^{s_2}, the OON node finds the rule r_o of Sl_o satisfied by m_v, it then forwards profile P_{Sl_v} to the next node. At the last propagation node n^{s_5}, the OON node then notifies m_v to Sl_o. Notice that if Sl_v arrives before Sl_o, the Sl_o will not be notified until Sl_v refreshes.

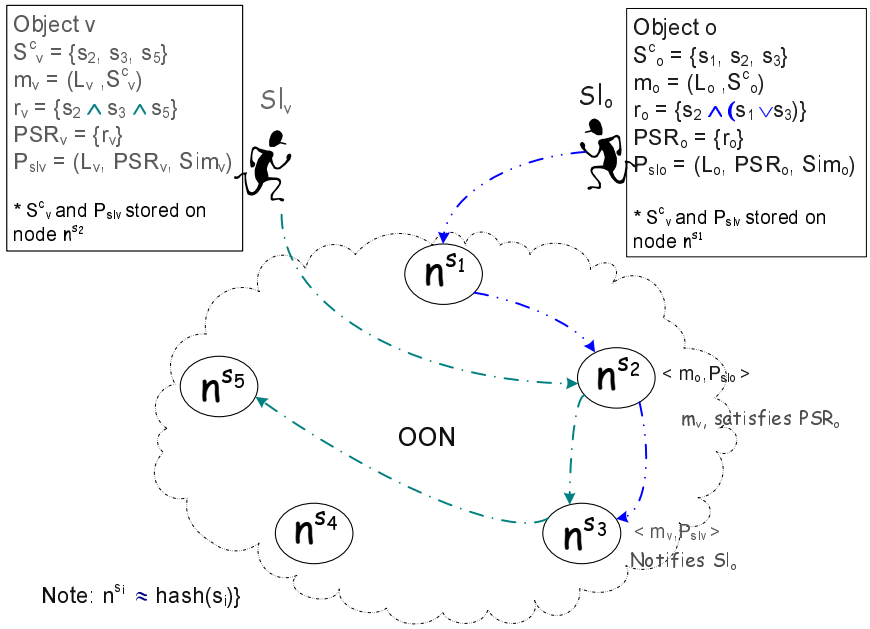

Figure 29: Crawler-based Scheme.

4.3.1 Costs and Benefits

Overall, the crawler-based scheme saves storage space at the cost of additional refreshing. Consequently, the frequency at which refreshing is performed plays a crucial role in the overall performance of the scheme. In an environment where objects are created frequently, the rate at which a semlet refreshes must be high. This high rate is necessary to ensure that a semlet discovers the newly advertised objects of interest. In static environments, however, excessive refreshing may unnecessarily result in potentially prohibitive overhead. Therefore, the rate of refreshing must reflect the tradeoff between the awareness of the newly advertised object and the communications overhead.

4.4 MINIMUM-COVER-RULE SCHEME (MCR)

The minimum-cover-rule (MCR) scheme leverages the benefits of the aggressive and the crawler-based scheme, while reducing its functional costs in terms of communications and storage. To achieve this goal, the MCR scheme computes a minimum set of nodes to ensure that all interested semlets are immediately notified when new objects are advertised, while minimizing the storage requirements and communications overhead.

The minimum set of nodes is computed from a data structure representing a minimum set of key IDs associated with rules, referred to as the *Minimum Rule-Cover-Node Set* (*MRS*). Formally,

$$MRS = \{(h, R_h)\}, \tag{4.1}$$

where h is a key ID and R_h is a set of rules stored at the node responsible for h. For example, the *MRS* of a rule set $R = \{r_1 : (s_i \wedge s_j), r_2 : (s_i \wedge s_k), r_3 : (s_j \wedge s_l \vee s_j \wedge s_k)\}$ is represented by $\{(h^{s_i}, \{r_1, r_2\}), (h^{s_j}, \{r_3\})\}$. This implies that rule r_1 and r_2 are assigned to be stored at the node responsible for h^{s_i} and rule r_3 is assigned to be stored at the node responsible for h^{s_j}. Therefore, the storage space used to store these rules and the bandwidth used to carry these profiles are minimized.

Finding a MRS for a given rule set is a two-step process that includes extracting "*independent*" rules from the rule set, and assigning rules to the most frequently appearing concepts. An *independent* rule is a rule which can be evaluated independently from other rules, and which cannot be broken down to smaller independent rules. In fact, an independent rule can be viewed as a Boolean expression that only has a "conjunctive ($\wedge$)" operator. Consider a rule set $R = \{r_1 : (s_i \wedge s_k), r_2 : (s_i \wedge s_m \vee s_m \wedge s_l)\}$; only r_1 is considered to be an independent rule. r_2 is not because it contains two conjunctive clauses; it can be broken down into two independent rules $(s_i \wedge s_m)$ and $(s_m \wedge s_l)$. An independent rule is used to identify a concept dependency, which enables selection of the minimum number of nodes necessary to store a given rule. For example, r_1, in the example above, can be stored at either n^{s_i} or n^{s_k}. The advertising and retrieval of this scheme is similar to that described for the aggressive scheme, except the query carries the MRS to direct where to store rules. The pseudo-code is described below.

Semlet: Starts object advertising and retrieval (given an object metadata $m_o = (L_o, S_o^c)$, and a semlet profile $P_{Sl_o} = (SL_o, PSR_o, SIM_o)$, where $PSR_o = \{r_o\}$, and $r_o = (S_o^u, \otimes)$)

```
1: //obtain routing keys RK_o
2: S_o ← S_o^c ∪ S_o^u
3: for all s ∈ S_o do
4:   adds hash(s) to RK_o
5: end for
6: computes the MRS_o for PSR_o (the details of finding a MRS are described below)
7: selects a key k_i from RK_o
8: composes a query query_o ← [k_i, RK_o, m_o, P_Sl_o, MRS_o]
9: sends the query to an OON node located within its domain
10: waits for the returned query or an advertised object metadata from other
    semlets
```

OON: Receives $query_o = [k_i, RK_o, m_o, P_{Sl_o}, MRS_o]$

```
1: if the OON node is not responsible for k_i then
2:   forwards the query to the neighbor that makes the most progress towards the
     node responsible for k_i
3: else
4:   //collect semlet profiles to advertise
5:   if the OON node is responsible for any concepts in S_o^c then
6:     retrieves the semlet profiles satisfied by the object metadata m_o
7:     stores and indexes m_o on the node
8:   end if
9:   //collect metadata of objects of interest
10:  if the OON is responsible for any concepts in S_o^u then
11:    retrieves the objects' metadata that satisfies the semantic rule r_o
12:    if the OON node is responsible for any key in MRS_o then
13:      stores and indexes a part of the semlet profile including the semlet locator
         and the semantic rules associated with the key defined by MRS_o on the
         node
14:      removes the member(s) of MRS_o associated with the key
15:    end if
16:  end if
17:  removes the keys that this node is responsible for from RK_o
18:  if no routing keys left in RK_o then
19:    advertises m_o to the semlets of interest
20:    returns all the collected object metadata to the querying semlet
21:  else
22:    selects a key k_j from RK_o
23:    forwards the query, the collected object metadata, and semlet profiles to
       the node's neighbor that makes the most progress towards the OON node
       responsible for the k_j
24:  end if
25: end if
```

Semlet: Computes a MRS for PSR_o

```
1: gets independent rules from PSR_o
2: counts number of appearances of each concept of the independent rules
3: sorts all the concepts ascendingly based on the number of appearances of the
   independent rules
4: goes through this sorted list of concepts from the top
5: while there is at least one independent rule left to assign more space do
6:   assigns the independent rule(s) to the concept that it appears on
7:   removes the rule(s) from the independent rule set
8:   moves to the next concept in the list
9: end while
```

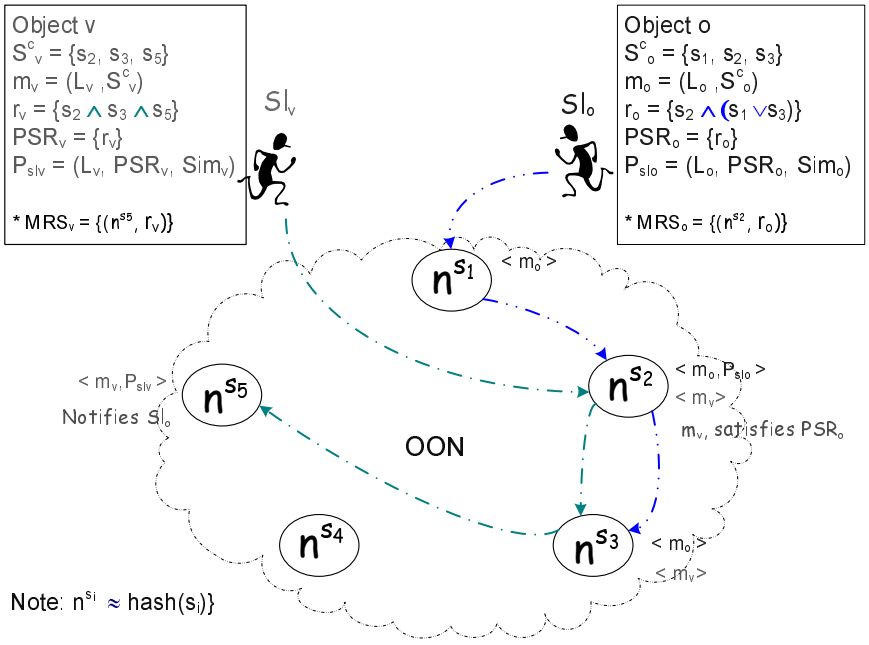

Figure 30: MCR Scheme.

Figure 30 illustrates the example scenario using this scheme. Similar to that using the aggressive scheme, a semlet Sl_o sends a query to propagate to nodes n^{s_1}, n^{s_2}, and n^{s_3}. These OON nodes also store the object metadata m_o. However, based on the MRS_o, only node n^{s_2} stores the semlet profile.

Note that rule r_o can be broken down to two independent rules: $s_2 \wedge s_1$ and $s_2 \wedge s_3$. Based on these rules, the number of appearances of s_2 in all the independent rules are the highest among the three concepts. Therefore, the rule r_o is assigned to n^{s_2}.

Later, the semlet Sl_v arrives at the OON. It then sends a query to propagate to nodes n^{s_2}, n^{s_3}, and n^{s_5}. At n^{s_2}, the OON node finds the rule r_o of Sl_o satisfied by m_v, it then forwards profile P_{Sl_o} to the next node. At the last propagation node, n^{s_5}, the OON node notifies Sl_o about m_v. Notice that, using this scheme, the order of the queries from both semlets does not affect the object notification. As soon as the object is advertised, the semlet is notified.

4.4.1 Costs and Benefits

This scheme minimizes the storage for personalized semantic rules and matching computation. Consider the case, when a semlet Sl_o has a rule set $R = \{r_1 : (s_i \wedge s_j), r_2 : (s_i \wedge s_k), r_3 : (s_j \wedge s_l \vee s_j \wedge s_k)\}$ and $MRS = \{(h^{s_i}, \{r_1, r_2\}), (h^{s_j}, \{r_3\})\}$. Compared to the aggressive scheme, the semlet sends rules r_1 and r_2 to be stored at n^{s_i} and r_3 to be stored at n^{s_j}, instead of sending all rules to be stored at $n^{s_i}, n^{s_j}, n^{s_k}, n^{s_l}$. Additionally, when another semlet advertises an object with semantics $\{s_i, s_k\}$ to n^{s_i},and n^{s_k}, the rule-checking against the metadata will occur only once at n^{s_i}, instead of multiple times on both n^{s_i} and n^{s_j}. Furthermore, the complexity of finding the *MRS* for a given rule set does not affect the effectiveness of semlet advertising and retrieval. The *MRS* can be computed *off-line* prior to semlet advertising or retrieval.

4.5 CONCLUSION

This chapter presented three object advertising and retrieval schemes to explore the effective and efficient mechanisms and protocols to support semlets in order to develop Personalized Webs. These three schemes have different benefits and drawbacks that have a great impact on the scalability and robustness of Personalized Webs. The first scheme, the *Aggressive* scheme, proposed to store and index object metadata and semlet profiles on all of the OON nodes responsible for their associated concepts. Using this scheme, multiple copies of object metadata and semlet profiles are stored on the OON. Consequently, the benefit of this scheme is the added availability of object metadata and semlet profiles. On the other hand,

the scheme has two main drawbacks. First, it increases the storage space required to store object metadata and semlet profiles on the OON nodes. Second, the OON nodes need to filter out the object metadata and semlet profiles found in multiple places when they perform object advertising and retrieval.

The second scheme presented, referred to as the *Crawler-based* scheme, reflects the opposite end of the spectrum from the aggressive scheme. The crawler-based scheme uses only a subset of OON nodes to store and index both the object metadata and the semlet profile. To avoid missing any object notifications, each query is refreshed periodically to renew discovery and advertising. Along with the benefits of a reduction in storage on the OON comes the need to increase the communications that perform refreshing. Arguably, this scheme is expected to perform well even in the churn environment of OON nodes. The experiment that investigates this topic will be presented in the next chapter.

The third scheme proposed, referred to as the *Minimum-cover-rule* (MCR) scheme, leverages the benefits of the aggressive and the crawler-based scheme, while reducing the functional costs in terms of communications and storage. The key to this scheme is the mechanism used to find the minimum set of nodes referred to as the *Minimum-Rule-Cover-Node set* (MRS) to store rules. The scheme exploits the dependency of concepts in a semantic rule to reduce the number of nodes that store the rule. For example, a rule comprised of a set of dependent, ontological concepts can be stored on a node responsible for one of these concepts. Since the advertising query related to these concepts would propagate to all the nodes responsible for the concepts, the OON node that stores the matching rule will eventually process the query. Therefore, a query does not require refreshing, while the storage on the OON nodes and the communications among the OON nodes are minimized. This scheme is expected to outperform the aggressive scheme and the crawler-based scheme.

To realize these three object advertising and retrieval schemes, a prototype implementation of the Personalized Web architecture was developed. To explore the performances of these schemes with regards to their effectiveness and efficiency in advertising and retrieval a set of simulation and emulation based experiments using this implementation were conducted. The details of the implementation and the experiments are discussed in the next chapter.

5.0 PROTOTYPE IMPLEMENTATION AND PERFORMANCE ANALYSIS

This chapter presents a prototype implementation and an experimental framework for the Personalized Web architecture proposed in the previous chapter. The prototype implementation serves to demonstrate a proof of concept of the proposed Personalized Web. The experimental framework evaluates the effectiveness and efficiency of the object advertising and retrieval schemes presented in Chapter four.

The chapter is organized into two main sections. The first part describes the prototype implementation. The second section explains the experimental framework, including the metrics used to evaluate the efficiency and effectiveness of the object advertising schemes, as well as the design and procedures of the experiments. The chapter concludes with the experimental results and a discussion about the performances of the schemes in various circumstances.

5.1 PROTOTYPE IMPLEMENTATION

This section presents a prototype implementation of the Personalized Web architecture, which is comprised of two main components: a semantically-aware semlet agent, and an ontology overlay network. This implementation is developed to assess the performance of object advertising and retrieval. It supports only a subset of the functions of the semlet and OON. The functions implemented for a semlet component include query generation to search and advertise objects of interest for the user, and the collection of the objects of interests advertised to the semlet. The OON's functions supported by this prototype include the overlay network management, and the functions necessary to support semlets to route and locate objects of interests advertised within the CoIs. The related functions omitted in this prototype include view development of the personal web of interest for each user and the optimization functions that improve the performance of the OON such as load balancing and replication.

To develop these components, the Bamboo-DHT software package is used[1]. The Bamboo-DHT is designed to be a scalable and robust DHT-based overlay infrastructure that openly

[1]Available at: http://www.bamboo-dht.org/

supports a public DHT service, namely *OpenHash*[2]. The Bamboo software package offers a set of application programming interfaces (APIs) that allow semlet and OON components to seamlessly implement the DHT-based routing and data location. Additionally, it includes the simulation tools and utilities, which can be exploited to evaluate object advertising and retrieval schemes.

The following section describes the implementation of the Personalized Web architecture based on the Bamboo-DHT. The section first introduces the overview of Bamboo-DHT and explains the integration of the Personalized Web components to the Bamboo-DHT components. It goes onto present the details of the implementation of the main components, semlet and OON.

5.1.1 Bamboo-DHT: A DHT Substrate of the Personalized Web Architecture

The Bamboo-DHT is a recently proposed DHT-based overlay substrate that aims at improving the robustness and scalability of the overlay network. It is known as a successor of PASTRY[78], one of the first generation DHT overlay architectures. Compared to PASTRY or other DHT overlay substrates, the routing algorithms and neighbor management algorithms are more incremental. The differences between these algorithms allow the Bamboo-DHT to better withstand large membership changes in the DHT as well as continuous churn in membership, especially in a bandwidth-limited environment[77].

The Bamboo-DHT software package is written in an event-driven, single-threaded programming style, which allows the Personalized Web components to be integrated with the DHT components in a scalable and seamless manner. Each component in the Bamboo-DHT is implemented as a *stage* which makes itself known to the system by registering with the core component, referred to as an *AsynCore*. The component also announces to the AsynCore a set of triggers for its actions through the types of events and messages. As such, the arrival of an event or message at the AsynCore causes the associated component to trigger its actions. This framework allows Personalized Web components to be incrementally added to the DHT-component without causing a change in the foundation of the software package.

[2]See the OpenHash project at http://www.openhash.org/.

Moreover, by implementing a single-tread programming style, the semlet and OON processes are simplified. At any point in time there is only one event or message triggering a stage. As a result, the Personalized Web components can be seamlessly integrated with the existing DHT components in a scalable manner.

To develop a Personalized Web architecture, two stages, namely the *semlet stage* and the *OON stage*, are implemented and integrated into the Bamboo-DHT software package. Figure 31 depicts the software architecture of the Personalized Web. The AsynCore component centrally manages all stages registered on it. Each stage maintains a queue of events with which it is associated. Once an event or a message arrives at a Bamboo-DHT node, it is sent to the AsynCore. The AsynCore then dispatches it to the associated stage(s). The event or message is put into the queue waiting for the stage to process.

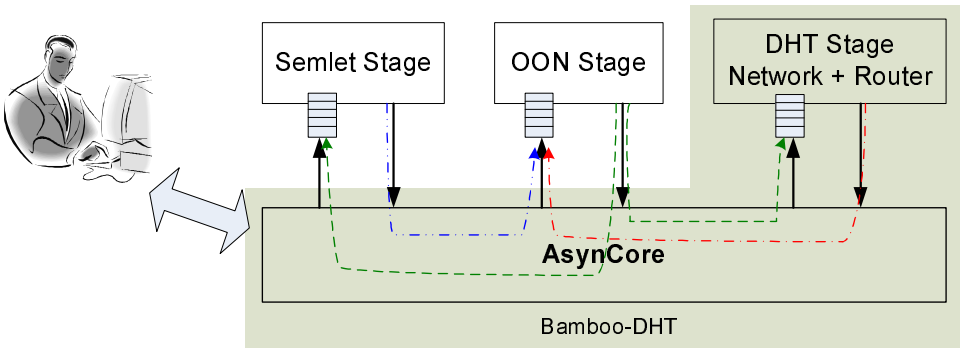

Figure 31: Personalized Web Stages.

Figure 31 shows three main stages: the DHT stage, the semlet stage, and the OON stage. The DHT stage implements router functionalities at the overlay layer level. It is responsible for routing the messages arriving from other nodes and from OON components, and for other overlay network functionalities including neighbor selection and management. The semlet stage interacts with the OON stage in response to the associated objects and personalized semantic rules. It is triggered by the event or message sent by the user. The OON stage handles events and messages that come from both the semlet stage and the DHT stage. The message from the semlet stage triggers the OON stage to compose the query necessary for routing. The OON stage then forwards the query to the DHT stage to send out to the node's

neighbor(s). The message from the DHT stage triggers the OON stage to store the data or, if the semlet is interested in the data, to deliver it to the semlet stage.

Based on the responsibility, the OON stage is the key for the scalability and robustness of the Personalized Web architecture. Object advertising and retrieval are centrally managed by the OON stage. It controls how the queries are sent and how the data are stored on the node. As a result, the efficiency of bandwidth and storage depends on the mechanisms the OON stage uses to perform object advertising and retrieval. The OON stage is developed to evaluate these mechanisms. The experimental frameworks for these schemes are presented in the sections that follow.

5.2 EXPERIMENTAL FRAMEWORK

In the previous section, the prototype implementation for the Personalized Web architecture was presented. This implementation aims to assess the mechanisms that the OON uses to perform object advertising and retrieval because these mechanisms impact largely the performance of the Personalized Web. This section moves on to discuss the experimental framework that evaluates these mechanisms. The assessment of the performance of these mechanisms is one of the main challenges for this research. The challenge stems from the complexity of setting up the underlying overlay network, which is normally composed of a set of machines whose number ranges from a few hundreds to thousands. To deal with this complication usually a simulated network is used in the experiments. However, using a simulated network may raise the question of whether the performance obtained from a simulated overlay network is consistent with that of a real network. Additionally, one may argue that the implementation of a simulation is not as effective when it is used on the real network.

In response to these arguments, the framework is comprised of two sets of experiments. The first set of experiments employs a network emulation, in which an emulator simulates the characteristic network connections, bandwidth and latency delay. The emulated network allows both OON processes and simulated semlet processes to interact in a real-time, network

environment. The real implementation of the OON and semlets can be used directly over the emulated network. As a result, these experiments demonstrate a proof of concept of the Personalized Web architecture. However, due to the limitations of network size and the difficulties of emulating network dynamics, the first set of experiments aims to evaluate the performance of the schemes for a small-size, stable network.

The second set of experiments targets the evaluation of the schemes performances on a larger network with different churns of OON nodes. The main objective is to analyze the performance of these schemes in a changing environment, where information about objects of interest is inconsistently available on the OON nodes. The second set of experiments employs a discrete event simulation, in which simulated events are executed sequentially in discrete simulation times. The simulation technique is used to reduce the complexity of the experiments caused by the concurrency of the processes. It also allows the joining and leaving of the OON nodes to be dynamically simulated without changing the setup or configuration.

The following sections explain the procedures of the emulation and simulation experiments. Prior to the explanation, the next section discusses two main performance metrics, namely effectiveness and efficiency. These metrics are used to evaluate the performance of the object advertising and retrieval schemes for all of the experiments.

5.3 EFFECTIVENESS AND EFFICIENCY METRICS

The Personalized Web architecture is designed with two performance objectives: scalability and robustness. To achieve these objectives, the Personalized Web architecture must enable users to discover and advertise objects of interest among members of CoIs in an effective and efficient manner. Consequently, the experimental framework employs *effectiveness* and *efficiency* to assess the object advertising and retrieval schemes.

The effectiveness of a scheme identifies how **accurate** and **complete** a discovery that the scheme provides. It is measured in terms of the *precision* and *recall* of advertising and retrieval[3]. The *precision* is used to measure the accuracy of object advertising and retrieval.

[3]The terms "precision" and "recall" used here are different from the metrics used in Information Retrieval

Formally, it is defined as:

$$precision = \frac{\text{number of relevant object metadata that are retrieved}}{\text{number of object metadata that are retrieved}} * 100 \tag{5.1}$$

The *recall* measures the capability to retrieve all relevant objects of the schemes. Formally,

$$recall = \frac{\text{number of relevant object metadata that are retrieved}}{\text{number of relevant object metadata in the system}} * 100 \tag{5.2}$$

The efficiency of a scheme identifies how much resources are used to support object advertising and retrieval. It is measured in terms of the network and peer resources including communication bandwidth and storage space used for advertising and retrieval.

5.4 EMULATION-BASED EXPERIMENTS

In order to demonstrate proof of concept of the Personalized Web, a network emulation tool is used. This tool provides the basis for testing real data traffic in a simulated network environment. Network emulation refers to the ability to introduce a network emulator into a live network. Special objects within the emulator are capable of introducing live traffic into the emulator and injecting traffic from the emulator into the live network.

Using the emulation, a network of a few hundreds nodes can be simulated with a small number of machines. Each machine simulates a set of processes for the nodes, and the connections of these nodes and the nodes simulated by other machines are provided through an emulator. The emulator emulates network delay by acting as a hub responsible for holding and releasing packets destined for different overlay nodes. The interactions received from these emulated machines can be considered as the interaction from the overlay network. A simple emulated network comprised of nodes from four network domains is shown in Figure 32.

research community.

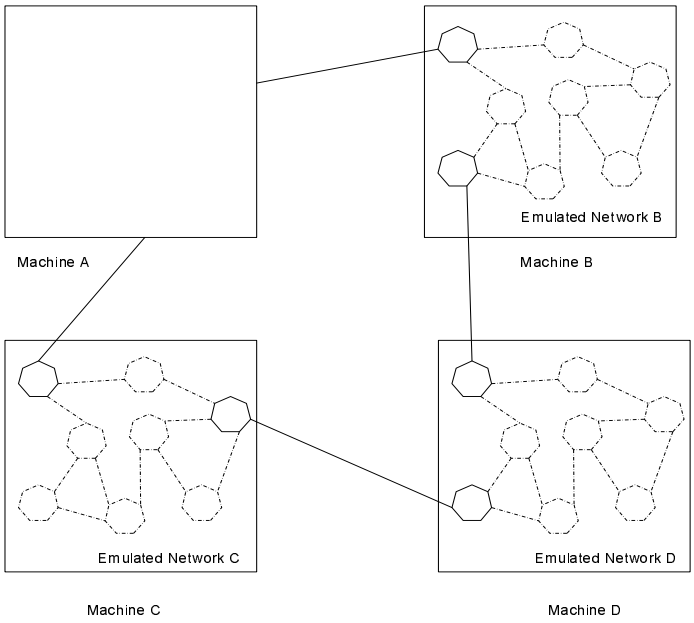

Figure 32: Network Emulation

5.4.1 Setup

The ModelNet emulator is used in these experiments [96]. With the help of the ModelNet emulator, different sizes of OONs are emulated using a cluster of sixteen machines connected with 100 Mbps links. For each experiment, each machine runs with an equal set of OON node processes which virtually connect to other OON node processes running either on the same machine or on other machines. One machine with a 2.0 GHz cpu is setup as an emulator, and the other 15 each with 1.4 GHz cpu are used to emulate multiple OON nodes.

The network topologies used in the experiments are generated by the Inet-topology software[102]. Four 5,000-node, wide-area AS-level networks are generated, with 50, 100, 150, and 200 overlay nodes respectively. The next section explains the procedures of the experiments. The overlay nodes and the core network are connected using 10 Mbps links with 100 ms latency. All stub-stub links were assigned 1 Gbps with 20 ms latency. Each OON node ran both the semlet stage and the OON stage to support semlet advertising and retrieval.

5.4.2 Procedures

Ten experiments are conducted for each simulation scenario. In each experiment, prior to object creation, an OON structure is formed, and remains stable through out the experiment. All experiments were performed using five thousands objects. Each object was randomly created by one of the OON nodes using a Poisson process with an inter-arrival time of 20 seconds per OON node. Each object was associated with 2 to 4 concepts drawn from an ontology described using 100 concepts. This relatively small-size ontology is used in order to ensure some level of similarity between different objects, which in turn allows the formation of different Personalized Webs.

The selection of a concept associated with an object and a rule followed the *Zipf* distribution, where the frequency of selecting a concept that is the i^{th}-most-frequently-used is approximately inversely proportional to i. This distribution is well known in the representation of semantic selection in natural languages. Consequently, few Personalized Webs are expected to have large collections of objects.

Once an object is created by an OON node, the OON service generates a single rule of a conjunctive clause involving all concepts generated for the object. For example, a rule $s_i \wedge s_j \wedge s_k$ is generated for an object metadata with semantics $\{s_i, s_j, s_k\}$. This rule is then associated with a semlet profile, which is sent to an OON node responsible for the advertising and retrieval rule.

The average precision and recall is obtained from post-simulation computation. Upon the completion of the simulation, the number of *objects retrieved* for each semlet is computed. The information of all semlets, their object metadata, profiles and objects of interest from all the OON nodes is collected and used to compute the number of *relevant objects*, and those of which are *retrieved*. These numbers are then used to compute the precision and recall as described above.

Four refreshing intervals for the crawler-based scheme, of size 200, 400, 600, and 800 seconds respectively, are considered. The goal is to study the impact of the interval size on the performance of the crawler-based scheme in comparison to other schemes. Table 2 provides a summarization of the main parameters in these procedures.

Table 2: Simulation Parameters

Parameters	Semlet Advertising And Retrieval Schemes		
	Aggressive	Crawler	Min-Rule-Cover
Refresh Interval (Seconds)	n/a	200,400,600,800	n/a
Number of OON Nodes	50, 100, 150, 200		
Number of Objects Created	5,000		
Number of Concepts in the Ontology	100		
Number of Concepts Associated with Each Object	2-4		
Object Inter-Arrival Time (Seconds)	20		

5.5 SIMULATION-BASED EXPERIMENTS

This section explains the simulation-based experiments, which aim to measure the effectiveness and efficiency of the object advertising and retrieval schemes in a churn environment. These experiments utilize the technique referred to as discrete event simulation, in which actions of the components in the system are encapsulated into events. Each event is processed sequentially based on its associated times.

The simulation-based experimental framework offers two main advantages over the emulation-based experimental framework. First, the simulation-based framework allows the experiments to be conducted with a larger scale OON because the simulation reduces the concurrency and complexity of the interactions of the components. The second advantage is the ability to dynamically change the OON members without changing the configuration in the experiment. This ability is needed to simulate the churn of the OON. The following sections describe how to setup the simulation-based experiments and how to conduct these experiments.

5.5.1 Setup

The simulation software provided with the Bamboo-DHT package is used for these experiments. Using this software changes slightly the implementation of the Personalized Web architecture, which is based on the Bamboo-DHT substrate. The network module in the DHT-stage is replaced by the simulated network module (refer to Figure 31 for more details). The software is setup on a machine with a 3.2 GHz cpu and 1 GB of ram.

The simulation-based experiment also uses a different network delay matrix. The experiments use an approach that focuses on ensuring the simulated network latencies follow the distribution of real network latencies on the Internet [43]. The network latencies between nodes are randomly sampled from the King dataset [22], which is extracted from real measurements of the round-trip times (RTTs) between 1,024 DNS servers.

5.5.2 Procedures

Most of the procedures used for these experiments are similar to those used in the emulation-based experiments. The only difference is the dynamic joining and leaving of the OON nodes at run-time. In other words, the OON structure is formed and dynamically changes as nodes randomly join and leave the network.

Nodes are classified into two classes: *long*-lived and *short*-lived. A long-lived node is characterized by a node that joins the network once and remains in the network throughout the simulation. While short-lived node is defined as a node that joins and leaves the network with an average lifetime of two minutes. This set of experiments simulates three scenarios that vary based on the percentage of node churn, 10%, 35% and 50%, on a 500-node network.

5.6 EXPERIMENTAL RESULTS

This section presents the experimental results from both sets of experiments. First, the section explains the results of the schemes effectiveness in terms of the precision and recall discussed in Section 5.3. The results are classified into results from the stable environment

and results from the churn environment. The following section discusses the efficiency results of the schemes in terms of bandwidth and storage.

5.6.1 Stable Environment: Precision and Recall

In the stable environment, where OON nodes are persistently available, the results show that the aggressive and minimum-rule-cover schemes give 100% precision and recall, while the crawler-based scheme only achieves 100% precision. These results can be explained by the fact that, using a crawler-based scheme, a semlet may miss objects that are advertised earlier and stored on nodes that the semlet does not visit.

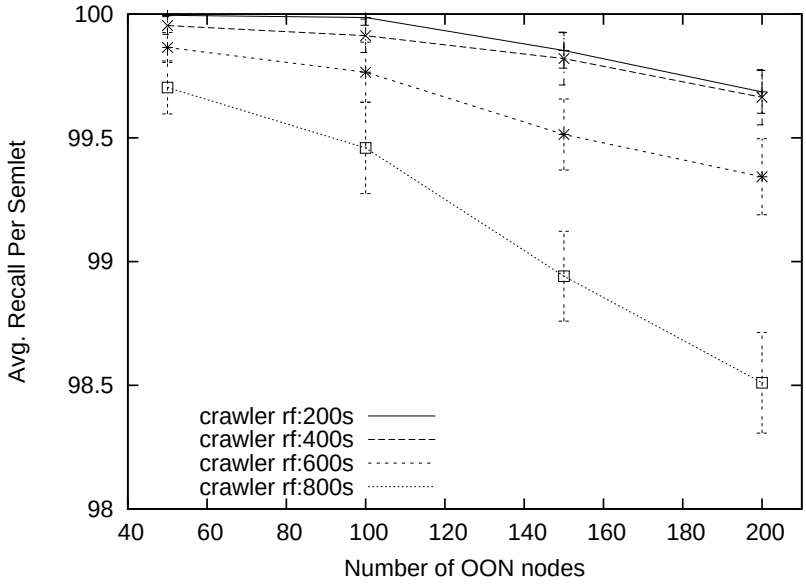

Figure 33: Recall vs. OON Size

In addition to the potential for semlets missing objects, the recall obtained by using the crawler-based scheme is directly impacted by the size of the OON and the size of the refreshing intervals. As illustrated in Figure 33 and Figure 34. As the number of OON nodes increases, the recall decreases. Similarly, as the refreshing interval increases, the recalls decrease. These results are explained further by the fact that the higher the number of OON nodes, the higher the chance that the semlet will miss an advertised object. Similarly, the

larger the refreshing interval, the greater the chance that the semlet could miss an advertised object.

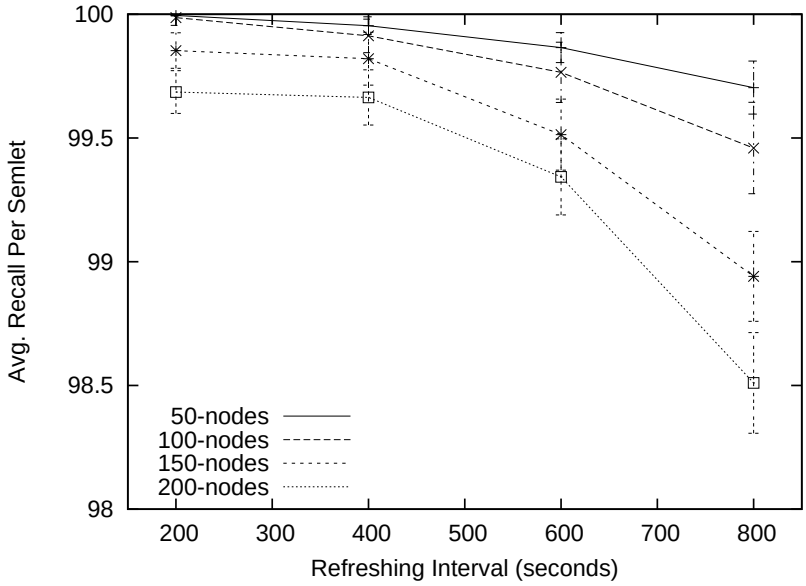

Figure 34: Recall vs. Refreshing Interval

5.6.2 Churn Environment: Precision and Recall

In the churn environment, the results show that the aggressive scheme and the crawler-based scheme with small refreshing periods (200 and 400 seconds) outperform the minimum-rule-cover scheme, especially when the percentage of churn nodes in the network is high. These results are illustrated in Figure 35.

The results, illustrated in Figure 36, also exhibit that, in the churn environment, as the refresh interval increases, the effectiveness of the crawler-based scheme is reduced considerably. This phenomena can be explained by the fact that the churn of the OON causes the loss of object metadata and semlet profiles previously advertised. The aggressive scheme is the most effective since it provides some level of replication. The crawler-based scheme, with small refresh intervals and without replication, outperforms the minimum-rule-cover scheme due to the fact that its refreshing strategy provides some level of data recovery. As

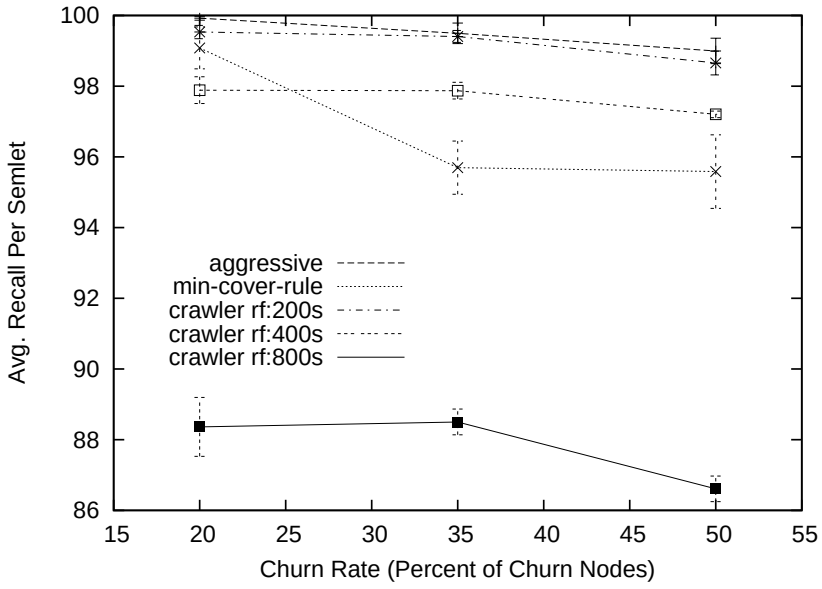

Figure 35: Recall vs. OON Size in a Churn Environment

the refresh interval increases, the effectiveness of the crawler-based scheme decreases. With its high refreshing interval, the data recovery cannot catch up on the data that is lost.

5.6.3 Bandwidth and Storage Efficiency

The results from both the simulation-based and the emulation-based experiments confirm that the crawler-based scheme consumes more bandwidth than the other two schemes. Figure 37 demonstrates the impact of this consumption in a stable environment.

The results also show that OON size does not impact the bandwidth consumption of the aggressive and minimum-rule-cover schemes. However, in the crawler-based scheme, as the OON size increases, the bandwidth consumption per node decreases. This trend can be explained by the fact that, when the OON is small, most of the OON nodes handle the refreshing traffic. On the other hand, when the OON increases in size, this traffic is spread widely across the OON nodes. As a result, the average bandwidth consumption decreases.

The results illustrated in Figure 38 show that the crawler-based scheme deploys storage the most efficiently of the three schemes. On the other hand, the aggressive scheme exhibits

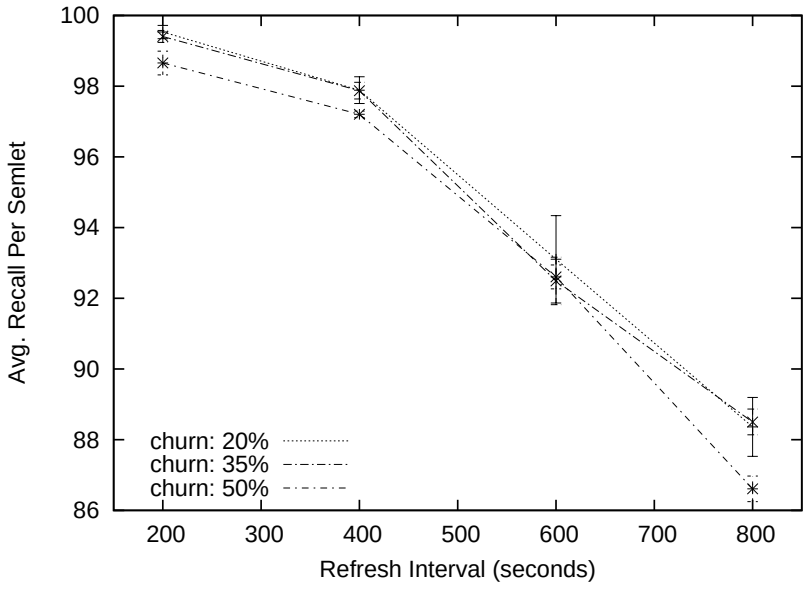

Figure 36: Recall vs. Refreshing Interval in a Churn Environment

the worst performance with respect to storage. The results also show that the average storage used per OON node decreases, as the OON size increases. These results occur because the metadata and the profiles are spread among more OON nodes as the OON's size increases.

5.7 RESULT ANALYSIS

The conclusions that can be made from the experimental results from both no-churn and churn environments are three folds. First, the aggressive scheme provides high retrieval recall in both churn and no-churn environment. However, using the aggressive scheme, semlets pay high storage costs and moderate bandwidth utilization. Second, even though the crawler-based scheme gives a low recall compared to other schemes in the no-churn environment, it tends to give better recall in a high churn environment. In addition, the crawler-based scheme could be used to save storage costs at a higher-usage communication bandwidth. Lastly, while the MCR scheme gives the best performance among three schemes in the no-

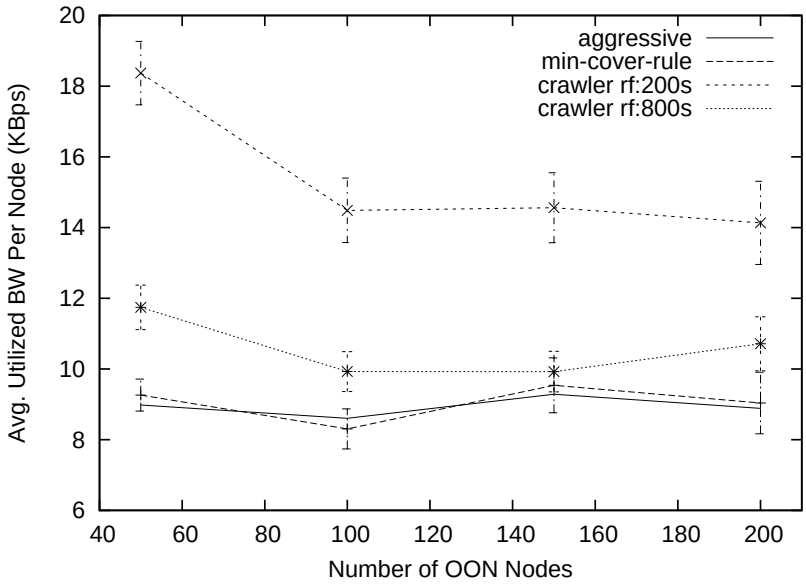

Figure 37: Bandwidth Efficiency.

churn environment, it exhibits that it could give a low recall in the churn environment. Table 3 summarizes the performance comparison of these three schemes.

To improve the performance of these schemes in terms of recall effectiveness, and bandwidth and storage efficiency, the key parameters of each scheme need to be chosen and implemented appropriately. For the aggressive scheme, the level of replication plays a crucial role for increasing the storage and bandwidth efficiency. In a stable environment, the number of replicated object metadata should be kept to a minimum since they are consistently accessible from the ORON. On the other hand, in the high churn environment, the level of replication should be high to allow semlets to access the object metadata even when there is high churn of the members of the ORON.

The key performance parameter of the crawler-based scheme is the refreshing interval. The crawler-based scheme could increase recall effectiveness, even in a high churn environment, by reducing its refreshing interval. In a stable environment, however, the refreshing interval could remain high to increase the efficiency of bandwidth utilization.

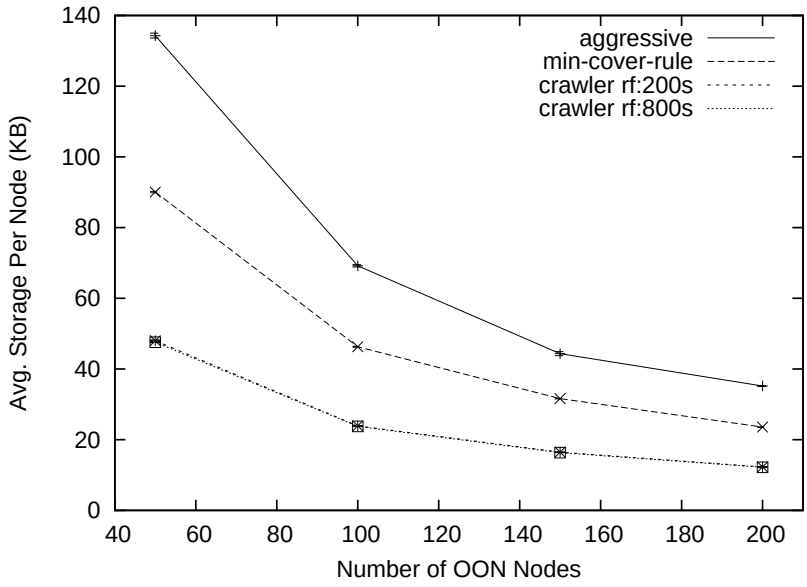

Figure 38: Storage Efficiency.

The key performance parameter of the MCR scheme is the level of replication. Similar to that of the aggressive scheme, in a stable environment the replication level of the MCR scheme should be kept at the minimum level for efficiency. On the other hand, in a high churn environment, the replication level should be increased to maintain a high recall. An alternative approach that could be used to improve the performance of the MCR scheme uses refreshing advertising and retrieval. Using this approach, the replication level could be kept at the minimum. Therefore, the MCR scheme with query refreshing could provide a high recall with high efficiency simultanously.

Future research could further investigate the amount of impact of these parameters. Appropriate values for the refreshing interval and the level of replication for the ORON in various environments should be found. An adaptive advertising and retrieval scheme that adjusts these key parameters over time in responding to the perceived environment should also investigated.

Table 3: Semlet Advertising and Retrieval Scheme Comparison

Scheme	**Effectiveness - Recall**		**Efficiency**	
	No Churn	**Churn**	**BW Utilization**	**Storage Costs**
Aggressive	High	Medium	Medium	Low
Crawler-based	Low	High	Low	High
MCR	High	Low	High	Medium

5.8 CONCLUSION

This chapter aims to achieve two objectives: demonstrate proof of concept of the Personalized Web through prototype implementation and assess the performance of the different strategies for object advertising and retrieval presented in the previous chapter. To achieve the first objective, the prototype implementation based on the Bamboo-DHT was developed. The Bamboo-DHT stages were integrated with the Personalized Web stages, namely the semlet stage and the OON stage, to provide the foundation of the Personalized Web architecture.

To achieve the objective of assessment, a set of emulation-based experiments and a set of simulation-based experiments were conducted. The results from the experiments show that in the stable environment the MRC scheme is more effective and efficient than the other two schemes. The MRC scheme exhibits 100 % precision and recall, while consuming an acceptable amount of bandwidth and storage. In the churn environment, however, the aggressive and crawler-based schemes outperform the MRC scheme. Due to the fact that the aggressive scheme provides some level of replication and the crawler-based scheme performs data recovery, they improve the availability of object metadata advertised in the OON. The refreshing interval plays a crucial role in achieving high recall for the crawler-based scheme. Compared to the aggressive scheme, the crawler-based scheme with a small refreshing interval

can achieve a higher recall. However, the crawler-based scheme incurs higher costs because it uses much more bandwidth.

In conclusion, selecting a scheme for semlets must consider the OON's environment, the recall objective, and the requirements for storage and communication. In a stable environment, the MCR scheme performs best. In a churn environment, the crawler-based scheme becomes a better choice. With requirements for greater storage and communication, the MCR may be better, even though a certain degree of recall degradation results.

6.0 PERSONALIZED WEB APPLICATION: "PERSONALIZED WEB SERVICE WORKFLOW"

This chapter examines how to implement a Personalized Web Service Architecture(PWSA), describing each fact of the problem in one of five sections. These explorations begin with an introduction to web services and web service workflow, followed by a discussion of the web service model and the challenges to enabling such a model. The third section describes the PWSA that addresses these challenges. The fourth section explains its main mechanisms, protocols and data structures. The chapter concludes with a discussion of future research directions.

6.1 WEB SERVICES AND WEB SERVICE WORKFLOW

Web services are expected to be the next generation of Internet-based applications that will dramatically change the use of the Internet [70, 49]. It is commonly referred to as "*a self-contained, self-describing modular application that can be published, located, and invoked across the Web*" [94]. It creates an opportunity for Internet-based applications to realize the automation of complex business tasks referred to as "workflows" [11, 32]. In the following section, the current infrastructure of web services is described, followed by a discussion regarding its ability to support workflow applications.

6.1.1 Web Service Technologies

Web services today evolve around four technology standards: *eXtensible Markup Language (XML)*, *Web Service Description Language (WSDL)*, *Simple Object Access Protocol (SOAP)*, and *Universal Description, Discovery and Integration (UDDI)*. These standards were developed to address the problems faced when trying to invoke a service across the Internet [6].

The first problem deals with *the syntax of a specification.* Web service specifications need a standard syntax that is flexible and extensible to define web service components, functionality, as well as an interaction protocol. XML is developing to serve such a purpose. It is commonly used as the syntax for web service specifications.

The second problem is *how to describe a service functionality, its offer and a way to invoke it in an efficient manner*. WSDL was developed to address this problem. It allows a service provider to describe functionalities of a web service in the form of a service interface and provides a way to invoke the service with a set of methods as well as the inputs and outputs of these interfaces. Additionally, it defines the network location of the web service, allowing consumers to remotely invoke the service over the Internet.

The third problem is *how to achieve an efficient and flexible protocol to access, and how to invoke a web service over the Internet*. SOAP was designed to address this problem. It defines a flexible message format and interaction patterns for web service invocation.

The fourth problem relates to *how to discover a web service in an efficient manner*. As the number of web services is expected to be large and their offers tend to be updated frequently, there is a strong need for a standard framework that allows users to discover the locations of services in an efficient manner. UDDI is developing to serve this need. It is developed as a standard framework for web service directory. It defines how to publish and locate a web service using a web service directory.

To further describe how these standards are used for web services today, the common web service model is illustrated in Figure 39. The Figure displays three main players involved in a web service: a service provider who provides the service, a service consumer who requests the service, and the UDDI directory that manages the service registry. To invoke a web service using the web service technology above, three steps are commonly employed. First, the service provider registers its service with a UDDI registry by sending the registry a WSDL-compliant service description encapsulated in a business entity, service and binding template. Second, the service consumer sends a discovery request for the service to the UDDI directory. The UDDI directory then returns a service interface that matches the request. Third, the consumer sends a service invocation request, which includes the service interface and the data required by the service. Upon receiving the request, the service then executes its task and returns the output to the consumer.

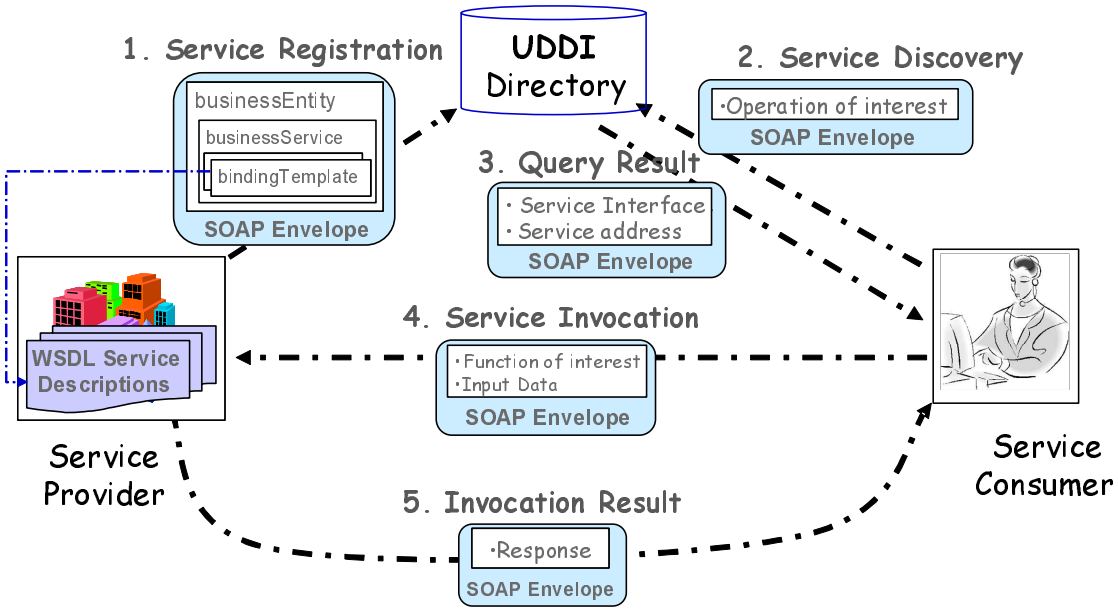

Figure 39: Web Service Model.

This web service framework provides several benefits. It offers a simplified mechanism to connect applications regardless of the technology, locations or devices they use. Second, it is based on industrial standards with universal support. Next, it leverages the Internet for the low cost communications. Last, it is self-describing, which then reduces the development effort needed to support automated processes, and reduces the collaboration effort between partners.

In addition to the web service framework, web services offer service consumers the potential to find and combine a set of web services to perform a complex task. Workflow theory can be used to automate the process to complete such a task. In the following section, the workflow theory is introduced, followed by a discussion on how it can be used to support web services.

6.1.2 Workflow

The automation of a "business process," in whole or part during which documents are passed from one participant to another for action, according to a set of defined rules is commonly referred to as the ***workflow***[1]. A participant in this context can be either a human resource or a computer application. A business process is defined as a set of interrelated tasks linked to an activity that spans functional boundaries [99]. A workflow, therefore, can be viewed as the implementation of a business process.

The main component of a workflow is the "*process logic.*" It defines a sequence of tasks and routing rules that must be followed, as well as deadlines and other business rules. Elements of a process logic and their relationships are represented by a "*process definition.*"

Several workflow modeling tools are proposed to model and analyze a workflow process. One popular tool is "petri-net" [65]. Its visual representation and ability to computationally analyze a workflow process make it a viable tool for a complex web service workflow. In the following section, the petri-net-based workflow modeling is introduced and further illustrated with an example.

6.1.2.1 Petri-Net Workflow Modeling A petri-net is a directed graph with an initial state called *initial marking*. Underlying it is a directed, weighted, and bipatite graph. A petri-net has two types of nodes called *places* and *transitions*. Places and transitions can be linked by a *directed arc*, which can either run from a place to a transition or a transition to a place.

Each transition has a certain number of *input places* and *output places*. A place p is considered an *input place* of a transition t when it has an arc running into that transition. Similarly, it is considered an *output place* of a transition when it has an arc running from that transition. A place with no incoming arc is considered as an *entrance* place. It is used to place a token at the *initial marking* state. On the other hand, a place with no outgoing arc is considered as an *exit* place. It is used to identify when the process finishes.

[1]Source: http://www.webasyst.net/glossary.htm

Tokens are used to define the *state* of the process. At the *initial marking* state, tokens are distributed to places in the graph defined by M_0. The movements of the tokens result in the state change of the process. Transitions are the active components that cause the movements in a petri-net. They move the tokens according to the following firing rule:

- A transition node is considered to be *enabled* when each of its input places contains at least one token.
- An enabled transition may fire, which results in removing one token from each input place and adding one token to each output place.

In petri-net workflow modeling, the concepts of *conditions* are represented by places, and *tasks* are represented by transitions [2]. A place containing a token is considered as holding the truth of the associated condition. It may also signal that a data item or resource is available for the connected task.

The formal definition of a Petri-net, taken from [65], is described as follows:

Definition 11. *A petri-net is a 5-tuple,* $PN = (P, T, F, W, M_0)$

$$
\begin{aligned}
P &= \{p_1, p_2, \cdots, p_m\} \textit{ is a finite set of places,} \\
T &= \{t_1, t_2, \cdots, t_n\} \textit{ is a finite set of transitions,} \\
F &\subseteq (P \times T) \cup (T \times P) \textit{ is a set of arcs,} \\
W &: F \to \{1, 2, 3, \cdots\} \textit{ is a weight function,} \\
M_0 &: P \to \{0, 1, 2, 3, \cdots\} \textit{ is the initial marking,} \\
&\quad P \cap T = \emptyset, \; P \cup T \neq \emptyset
\end{aligned}
$$

The basic graphical representation of a petri-net is shown in Figure 40. In the figure, there are two conditions a and b, represented by circles. There is also one task, J, represented by a rectangle. A token is denoted by a circle dot. It is marked in the place a, which indicates that the pre-condition for task J is set to true. As a result, the task J is considered *enabled*, or ready to execute. Once the task is completed, the token from place a is removed and a token is added to place b.

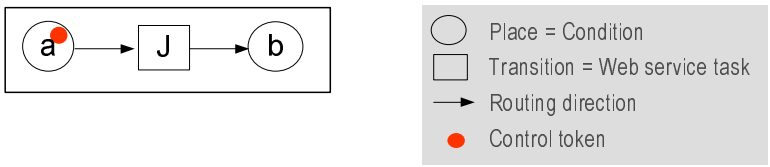

Figure 40: Basic Petri-Net Workflow Model.

Several workflow patterns are used to model how a token chooses routes in a workflow graph. Five basic workflow patterns include *Sequence*, *AND-Join*, *AND-Split*, *OR-Split*, and *OR-Join*. They are briefly discussed as follows:

- *Sequence* pattern defines the basic control flow, in which a workflow process is routed from node to node sequentially.
- *AND-Split* identifies a "parallel" control flow, in which a workflow process is split into multiple processes, one for each outgoing arc.
- *AND-Join* defines a "synchronized" control flow, in which a workflow process can only execute a task when all the conditions from all incoming arcs to the task are set to true.
- *OR-Split*, on the other hand, represents a "choice" control flow, in which a workflow process has choices in selecting the next task to execute.
- *OR-Join* defines the "any" control flow, in which a workflow process can start executing a task after one of the processes from incoming arcs pass the control to it.

These workflow patterns are illustrated in Figure 41.

6.1.3 Petri-Net Workflow Example

To demonstrate the petri-net workflow modeling, consider the case of a workflow for finding the best deal (i.e. lowest price) for a three-day trip for one person from Pittsburgh to San Francisco from August 1^{st} to 5^{th}, with the following preferences, and constraints:

- The user needs a round-trip airplane ticket, a hotel room, and a rental car,
- The number of connections of air travel must be no more than two,

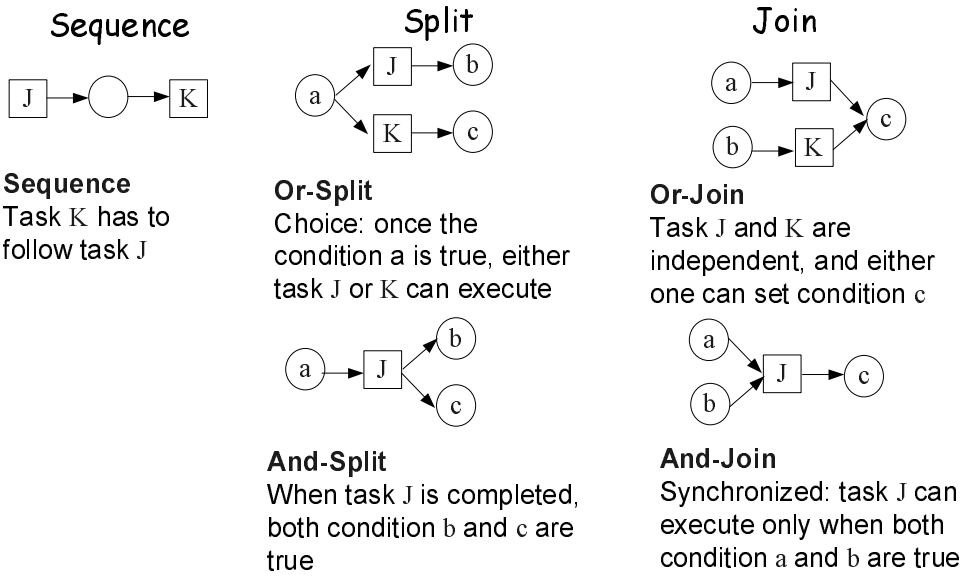

Figure 41: Workflow Patterns.

- The hotel must be rated at least three stars from the AAA company,
- The size of the rental car must be at least full-size, and
- The total budget must be less than one thousand dollars.

Given the above requirements and preference, a simple workflow process may have a process logic which executes the following steps:

1. Queries the travel agencies that provide reservation services for a round-trip airplane ticket, a hotel room, and a car rental in San Francisco.
2. Queries the agencies returned from the first step for the prices of available airplane tickets, hotel rooms, and car rental based on the above preferences and constraints. To increase effectiveness in the query, the workflow process is split into three processes to execute the following tasks in parallel:
   a. Queries the airplane ticket agencies, the prices and the information of available round-trip air plane tickets with the following constraints: departure: August 1^{st}, return: August 5^{th}, and at most two connections for each trip.

   b. Queries the hotel agencies, the prices and the information of the hotel room with the following constraints: check-in date August 1^{st}, check-out date: August 5^{th}, and rated three stars from AAA.
   c. Queries the car rental agencies, the prices and the information of the car rental with the following constraints: pick-up date: August 1^{st}, return date: August 5^{th}, pickup airport: SFO, and car-size: at least full-size.
3. Computes, after the agencies return all information from all tasks, the total prices of all combinations of the airplane tickets, hotel rooms, and rental cars that satisfy the time and price constraints. At this step, all split processes join together to progress to the next step.
4. Displays "Not Available" and finishes the process, if there is no trip that costs less than one thousand dollars. Otherwise, it follows the next steps.
5. Ranks these combinations based on the prices.
6. Selects the combination that gives the cheapest price.
7. Makes the reservations for the selected combination. At this step, the workflow process is again split to three processes to execute the following tasks in parallel:
   a. Queries the airplane ticket agency to buy the airplane ticket for the selected combination.
   b. Queries the hotel agency to reserve the hotel for the selected trip.
   c. Queries the car rental agency to reserve the car for the selected combination.
8. Finally, once all reservations are completed, prints the trip summary for the user.

Figure 42 shows the petri-net for this workflow. It starts with an entrance place with an outgoing arc to a transition associated with the first task. This transition is then connected to three output places with the And-Split pattern. As a result, three query tasks are spawned to execute in parallel once the first task finishes. Next, all of these tasks join at the next task with the And-Join pattern. The joining reflects a synchronized control flow, which forces the next task to progress if and only if all three input places acquire at least one token. The petri-net continues with a task sequentially connected. It then has an Or-Split to choose the next task based on the condition. The condition of *price > 1000 or price is not available*

leads to the display "Not Available" task. The other condition leads to an And-split to spawn the reservation tasks and an And-join prior to printing the trip summary at the end.

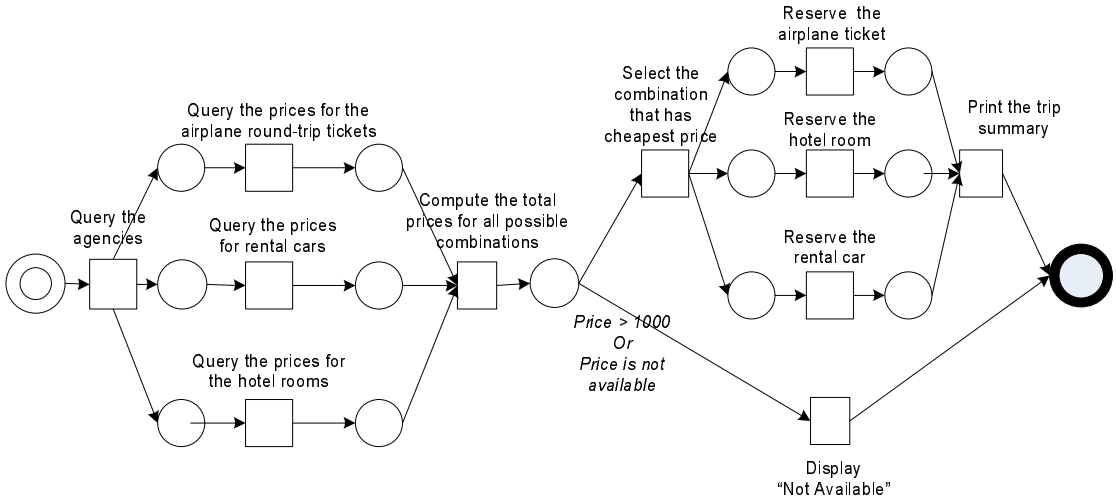

Figure 42: A Petri-net Workflow Model Showing the Planning Trip Process Logic.

6.2 PERSONALIZED WEB SERVICE WORKFLOW

The workflow theory provides a viable solution for web service automation on the next generation Internet. It allows a web service process to automate a sequence of web service tasks, that are defined by process logic to achieve a given objective. This automation is referred to as a "***web service workflow***."

The applications of web service workflow can be expected to be one of the most rewarding applications of the web service. With the ability to automate tasks, web service workflows allow organizations and individual users to achieve a set of tedious, repetitive, and time-consuming web service tasks without human intervention. They enable a composite business process which requires a pattern or series of interactions with web services.

This thesis focuses on the web service workflow for individual users, referred to as "***Personalized Web Service Workflow***.". It is defined as "*the process that facilitates a service consumer to complete a set of web service tasks to meet the consumer's personal objectives, and preferences.*"

A personalized web service workflow is modeled as a two-phase process. In the first phase, referred to as "*workflow specification*," the process generates a workflow logic definition from given workflow requirements and user preferences. In the second phase, referred to as "*workflow execution*," the process carries out the workflow logic obtained from the first phase. First, the process logic is decomposed into a set of web service tasks. Based on the process logic, the process then executes a sequence of the web service tasks. Finally, the results collected from all tasks are composed and then returned to the consumer. The model of this process is illustrated in Figure 43.

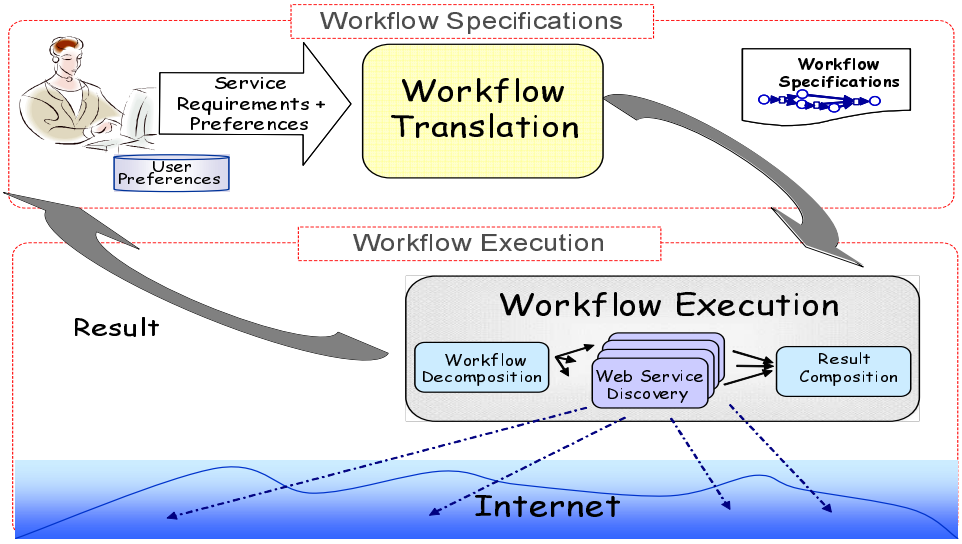

Figure 43: Personalized Web Service Workflow Model.

Various methods and frameworks for generating workflow definition from a set of requirements have been recently proposed in workflow and artificial intelligent(AI) research. In workflow research, several works proposed means to generate a specification of a process

model for a workflow [15, 84]. In the AI research, the works related to AI planning propose dynamic composition methods to generate a plan automatically [67, 58, 61, 104].

Only a few works related to workflow execution phase, however, have been proposed. They mostly focus on inter-enterprise business applications under a controlled and closed environment (i.e., within the same organization or under a single authority)[30, 105, 98]. Moreover, no work aims to support web service workflow for individual users.

To this end, the architecture including data structures, protocols, and mechanisms to support workflow execution is the main focus of the rest of this chapter. Prior to a discussion of this architecture, the challenges that need to be addressed by this architecture are discussed.

6.2.1 The Challenges of Enabling Personalized Web Service Workflow Execution

Enabling a personalized web service workflow in the open web environment poses several challenges. They stem from the autonomy and decentralization of the execution, as well as the increasing number and the dynamism of web services. These challenges are listed below:

- The increasing number of web services: the number of services available over the web has been increasing dramatically over the last couple years, and one can expect to have a huge web service repository search. A workflow process needs to be able to locate the service quickly and efficiently.
- Dynamism of web services: web services can be created and updated on the fly, thus the composition system needs to detect the updating at runtime and the decision should be made based on up-to-date information.
- Web service composition and orchestration: web services, once discovered, must be composed and orchestrated to fulfill a workflow specification. A workflow process must be able to handle the semantic and structural differences of service interfaces, and orchestrate the service invocation based on a given workflow.

- Workflow Monitoring and error handling: since no central authority to regulate and monitor the control and data flow among web services exists, a workflow process must be able to monitor its own activities running across the Internet, and it must be able to handle any errors once they are detected.
- Heterogeneity of web service ontologies: different organizations use different concept models to describe services. No unique language exists to define and evaluate web services in an identical manner. A resolution of structural and semantic differences among service providers and consumers must be supplied.

Note that the issues related to web service semantics, their interoperability, extensibility, and reusability are well addressed by the recently proposed *Semantic Web Services* vision [60]. Its goal is to make web services semantically-expressive. As such, machines in the form of agents can understand and interact with web services in an autonomous fashion. The key mechanism is to associate web service components with an expressive concept model, referred to as an ontology. With ontology, the semantics of input and output, and functionalities of web services can be made easy to understood. As such, they enable the automation of data and control flow among web services.

The proposed architecture, referred to as the "***Personalized Web Service Architecture***" is inspired by this vision. The main components and mechanisms to address the challenges above are discussed in the next section.

6.3 PERSONALIZED WEB SERVICE ARCHITECTURE (PWSA)

The basic design goal of the PWSA is to provide service consumers with the ability to adaptively execute web service workflow to meet their personal interests in an effective and scalable manner. To achieve this goal, the proposed approach augments the functionalities of a Personalized Web architecture with the capabilities of an agent-based technology to provide a flexible and scalable architecture capable of addressing the challenges discussed above.

The main component of the architecture is the ***"Ontology-based Registry Overlay Network" (ORON)*** and a semantically-aware workflow agent called a "***flowlet***." ORON acts as a rendezvous point between service consumers and providers. Its underlying infrastructure is a structured DHT-based P2P overlay network coupled with a service ontology providing semantically-based search for web services. Note also that the ORON is a supplement of the current web service infrastructure. It allows web services and UDDI directories to be discovered and advertised in a highly scalable and robust manner within an open web environment.

A flowlet, on the other hand, acts on a user's behalf to execute a web service workflow by performing web service orchestration through ORON. The overall workflow execution model using ORON architecture is illustrated in Figure 44.

As shown in the Figure, the ORON formed by a group of nodes acts as the rendezvous infrastructure between web service providers and web service consumers. On the one hand, it manages web services information advertised by providers while distributing the web service information among ORON nodes based on the ontotologies. On the other hand, it allows flowlets acting on consumers' behalf to find and orchestrate these web services to achieve web service workflows.

Note that in PWSA, providers and consumers are assumed to use ontologies to describe web services. In addition, providers and consumers, who share similar interests, agree on the ontologies they use, and are considered members of the same community of similar interest referred to as "CoSI." Consequently, through ontologies, providers and consumers can choose CoSI to advertise and discover web services.

To further discuss PWSA, the framework and models of the architecture are described in the next section. The mechanisms and protocols to help both providers to advertise web services and consumers to achieve personalized web service workflows are explained in the next section.

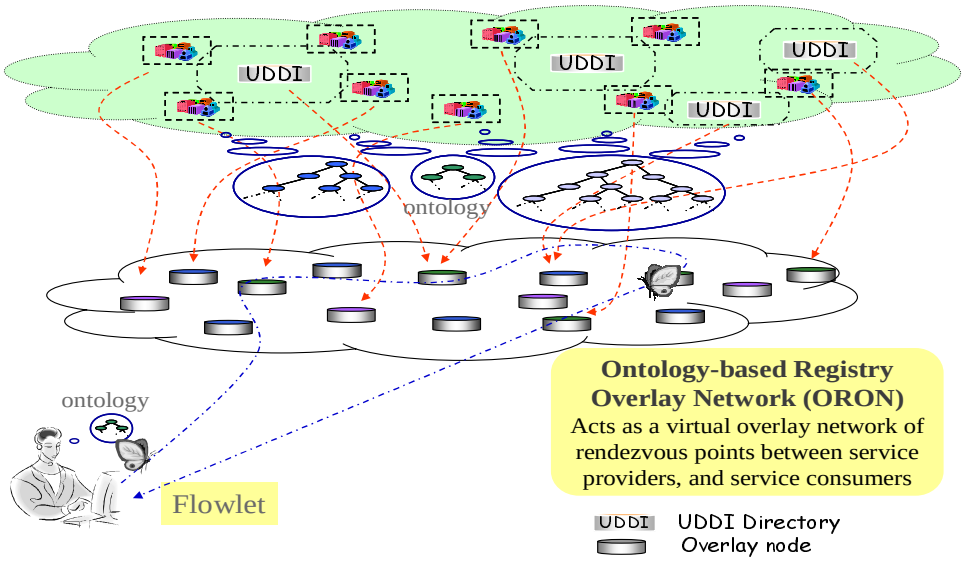

Figure 44: Ontology-based Registry Overlay Architecture.

6.4 FRAMEWORK AND MODELS

In this section, the framework of the PWSA will be described. First, the descriptions of the main components, their basic functionalities, and their data structures will be explained. After that the discussion focuses on the mechanisms and data structures used to form the ORON.

6.4.1 Service Ontology

In PWSA, semantics used to index, search, and discover services are represented by a set of concepts defined in an ontology referred to as a ***Community Service Ontology (CSOnt)***. A service ontology can be viewed as a dictionary of service categories and their relations, formally defined and used within a community. It is used to ensure consistency in semantic classification among services within the community.

In fact, various CSOnts can be expected to be created and distributed for users to reuse in the near future. One can predict the use of existing service ontologies such as *eTom* for the

telecommunications industry [35], *OTA* for travel industry[4], *OFX* for financial industry [29], etc. A new CSOnt may be derived from a service and product classification, such as *North American Industrial Classification System (NAICS)*, and *United Nations Standard Products and Services Codes (UNSPSC)* [45].

6.4.2 Service Profile

Each web service is associated with a service profile. A service profile is composed of three main attributes: "*location*," "*textual description*" and "service semantics." The *location* identifies where to access the service. It is represented by a URL composed of the host's IP address and port, and the name of the service. The *textual description* describes service name, the description of the service and other text-based descriptions. The *service semantics* identify the type, class, or category of a service, which is represented by a concept, or a set of concepts defined in a CSOnt. It may be also represented by functional semantics represented by the IOPE (Input, Output, Precondition, and Effects) model [57].

Definition 12. *Let sp_A be the profile of service A, and S be a set of service categories defined by CSOnt within the CoSI, then*

$$\begin{aligned} sp_A \;=\; & (url_A, V_A, S_A) \text{ where} \\ & url_A = \text{ its location} \\ & T_A = \text{ its textual description} \\ & S_A = \text{ its service semantic, and } S_A \subset S. \end{aligned}$$

Note that ontology-based description languages such as OWL-S [57] and WSMO [27] can be used to represent the service profile. They offer description languages for web services, which allow both the functional and the non-functional properties of a service to be described by semantics defined in the form of an ontology.

6.4.3 Web Service Task

A web service task is referred to as a unit of work whose objective is to find the best web service result with respect to the user's constraints and goals. Generally, to achieve such a task, the workflow process requires a service discovery and a service invocation. A workflow process needs to identify the web service to execute, the input data to send in order to execute, and the expected output data from executing the web service.

To support these functionalities, each web service task is characterized by a web service category defined by the CSOnt, the service constraints used to identify the web service, the input data used for service execution, and the output data expected from the service execution.

The service semantics, together with constraints are used to discover the best matched web service. The input data represented by a set of parameters and their value pairs is used to invoke the chosen web service. The expected output data defines the expected output values after invoking the chosen web service. The constraints are represented by a set of rules. For instance, with the constraint, "*a service must be certified by* Verisign," the web services that are not certified by *Verisign* are filtered out from the search.

Definition 13. *A web service task t is characterized by*

$$
\begin{aligned}
t \;=\; & \{S_c, IV, O, R\} \textit{ where} \\
& S_c = \textit{ a set of service semantics} \\
& IV = \textit{ a set of inputs and their value pairs} \{s_I^0 : v_I^0, ..., s_I^n : v_I^n\} \\
& O = \textit{ a set of expected output parameters} \{s_o^0, ..., s_o^k\} \textit{ and} \\
& R = \textit{ a set of constraints.}
\end{aligned}
$$

6.4.4 Web Service Workflow

The main component used to enable an automation of a series of web service tasks is the *web service workflow*. It is used to control a sequence of web service invocations, and the data flow among web service tasks. It is characterized by "a set of tasks," "a set of control and

data flow among these tasks," and "a set of required data input and expected data output for each task."

Inspired by the petri-net-based workflow model introduced in Section 6.1.2.1, the web service workflow is modeled as "*a directed graph*," in which places represent conditions, transitions represent web service tasks, and arcs represent the potential control and data flow among them.

Definition 14. *Let H be the workflow graph of workflow W, and Dat be a set of data required for the workflow execution. Furthermore, let H be characterized by a set of web service tasks T, let C be a set of conditions, and let E be a set of connecting arcs. Then,*

$$
\begin{aligned}
W &= (H, Dat), \textit{ where} \\
H &= (T, C, E) \\
&\quad T = \{t^0, \cdots, t^n\} \textit{ where } n > 0 \\
&\quad C = \{c^0, \cdots, c^m\} \textit{ where } m > 0 \\
&\quad E = \{(t, c) \cup (c, t) \mid t \in T, c \in C\} \\
Dat &= \{I_u \cup O(E)\} \\
&\quad I_u = \textit{ a set of user preferences and constraints} \\
&\quad O(E) = \textit{ a set of output obtained by executing task } e \in E, \textit{and} \\
&\quad \forall e^i \in E, IV^i \subset Dat
\end{aligned}
$$

6.4.5 Personalized Web Service Workflow Model

PWSA allows service consumers to express their interests in service execution and to automate web service workflow execution using ***Personalized Workflow Profiles (PWP)***. Three components comprise a PWP: "a set of goals" expressed in term of objectives and constraints, a "workflow specification" characterized by a workflow W defined above, and "a set of optional discovery properties" to define discovery policy.

Goals represent the overall objectives that the user wants to achieve from executing the whole workflow. Examples of such goals include "lowest cost," "quickest service," and "best rating service." A goal is represented by a "multi-parameter" object function ($Op()$) such as the maximize and minimize functions.

Discovery properties are used to specify the discovery policy. Two main discovery properties are *scope* and *time*. *Scope* is used to define the scope of the discovery. It can be defined as a DNS domain, such *.com* or physical domain such as US[2]. *Time*, on the other hand, defines the period of the discovery. It may be set as a one-time type of discovery, or as a persistent type of discovery within a specified time period. In the latter case, the PWP is stored on the *ORON* waiting for notification of a new service advertisement.

Definition 15. *Let P_f be a personalized workflow characterized by a set of goals(G), a set of discovery properties (D), and a workflow specification (W). Then,*

$$
\begin{aligned}
P_f \;=\; & (G, W, D) where \\
& G = \{op_0(), \cdots, op_n())\}, n > 0 \\
& W = \textit{ the workflow specification} \\
& D = (Sc, Time) \\
& \textit{ where } Sc = \textit{the discovery scope and} \\
& Time = \textit{the time period of the discovery}
\end{aligned}
$$

6.4.6 Workflow Agent: Flowlet

In PWSA, each personalized workflow profile is associated with a semantically-aware agent, referred to as ***flowlet***. It acts as the "surrogate" agent to execute a web service workflow process for its user. Given a *personalized workflow profile*, it discovers from ORON the best web services suitable for the tasks, and orchestrates these web services to achieve a given goal. This process is illustrated in Figure 45. Note that the flowlet is represented by the butterfly in the Figure.

[2]Applicable only to the web services whose location information is registered with ORON

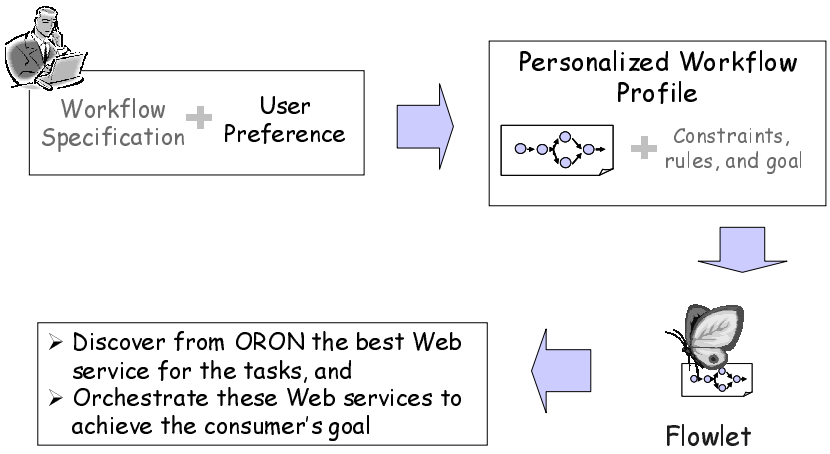

Figure 45: Service Acquisition Through A Personalized Workflow.

6.4.7 Ontology-based Registry Overlay Network(ORON)

The PWSA supports a user to discover, and orchestrate web services through the registry metadata network, referred to as an *Ontology-based Registry Overlay Network (ORON)*. The ORON uses the service ontology coupled with the DHT to infer a *semantic structure* that regulates location and access to a distributed set of service profiles and personalized workflows. Consequently, the ORON provides an adaptive and scalable, distributed search, and self-organization of data to support large scale, personalized workflow executions.

The ORON assumes the following:

- The Internet is organized into a set, $\mathbf{D}$, of autonomous domains. Each domain, $d \in \mathbf{D}$, elects a node, n_d, to act as a *registry node* in the registry farm.
- The set of registry nodes, referred to as $\mathbf{N}$, forms the basis of the registry farm. Hence, $N = \{n_d | d \in \mathbf{D}$ and n_d is an registry node$\}$. Each node, $n_d \in \mathbf{N}$, has a set of neighbors, N_d, which it uses to route data using a DHT-based strategy.
- The nodes in a given domain, d, address the search and advertising to the registry node $n_d \in \mathbf{N}$.
- Let $\mathbf{S}$ represent the set of conceptual services defined in CSOnt. Each registry node $n_d \in \mathbf{N}$ is assigned a set of service semantics $S^n \subset S$, and has the responsibility to

store and index service profiles, advertise services, and retrieve services of interest for the flowlet associated with any service semantic $s_i \in S^n$. For all $s_i \in S^n$, the node n_d responsible for s_i is the node whose $id \approx hash(s_i)$. A node that is responsible for s_i is denoted as n^{s_i}.

Note that ORON is a special case of an OON, which is designed to manage service profiles. The assumptions above are similar to those defined in section 3.2.3.

6.5 PWSA MAIN MECHANISMS

This section describes two main mechanisms that enable a user to achieve a personalized web service workflow using the PWSA framework and model. The first mechanism relates to how to facilitate service providers to advertise and register their web services with ORON, and how to make these services available to the consumers in a CoSI in an efficient manner. This mechanism is referred to as "Service Registration and Advertising." The second mechanism relates to how a flowlet discovers and orchestrates web services registered on ORON to achieve a set of of constraints and goals. This mechanism is referred to as "Flowlet-based Workflow Execution." Both mechanisms are discussed in more detail in the sections that follow.

6.5.1 Service Registration and Advertising

To make a web service available to service customers, a service provider needs to register the web service with the ORON. To do so, first, the service provider submits the web service profile to one of the ORON nodes. Upon receiving a service profile, the ORON node then uses the DHT-based strategy to route the web service profile to the ORON node responsible for its associated service semantics. The service semantic is used as the routing key on ORON. The targeted ORON node, upon receiving the web service profile, then stores and indexes the service profile with its associated semantic. Next, the ORON node notifies the service provider to allow the service provider to monitor the registration process. The process

continues until all of the ORON nodes responsible for all other service semantics complete profile storing and indexing.

To handle the dynamic nature of web services, and the churn of the network, service providers further refresh or update their service profiles on ORON periodically. Additionally, each the ORON node implements *timestamps* to detect unavailable or out-of-date services. The services without updates in a certain period of time can then be detected and removed to provide a consistent and up-to-date view of the web service registry. The detailed procedure that demonstrates service advertising and registration is described in the pseudo-code below and illustrated in Figure 46.

Service Provider: Given a service profile $P_A = (url_A, T_A, S_A)$

```
1: for all s ∈ S_A do
2:   Send P_A and its policies through DHT to the ORON node responsible for
     hash(s) to store and index with the service semantic s,
3:   Monitor and control messages to make sure that P_A reaches ORON node
     responsible for hash(s)
4: end for
5: Setup timer to repeat the service registration.
```

ORON Node: node n_i **receives the** P_A **associated with semantic** s

---

```
1: if n_i is not responsible for s then
2:   routeDHT(s, P_A)
3: else
4:   if P_A exists in the local registry then
5:     update the timestamp associated with P_A
6:   else
7:     store and index P_A with semantic s and the current time as the timestamp
       in the service registry,
8:   end if
9: end if
```

6.5.2 Flowlet-Driven Workflow Execution

In PWSA, a flowlet is used to carry out a web service workflow. It controls the workflow process causing it to execute a sequence of web service tasks in order to achieve a certain set of goals. The flowlet interacts with the ORON to discover web services and to perform web service executions.

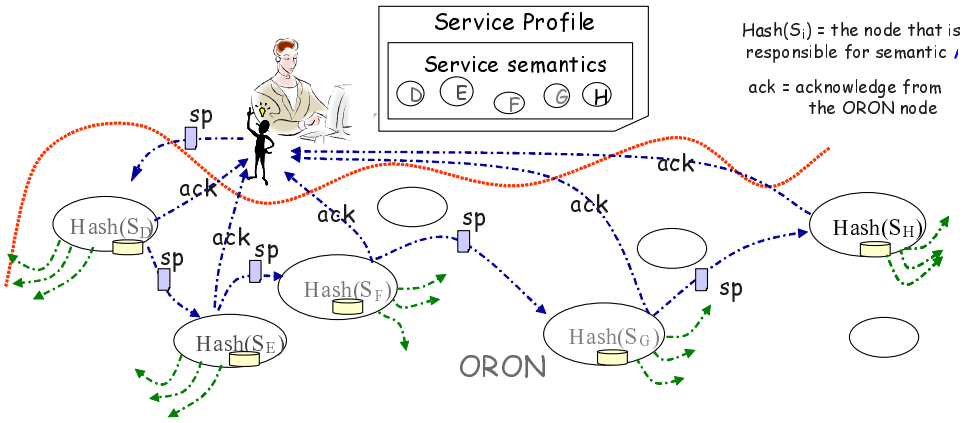

Figure 46: Service Registration and Advertising.

Based on how the ORON is used to support a web service workflow, two schemes are proposed. The first scheme, referred to as the *client-based workflow execution*, uses the ORON to discover web services to support a workflow execution. A flowlet performs a workflow orchestration, including web service executions and state update, at the client node. On the other hand, the second scheme, referred to as the *proxy-based workflow execution*, utilizes the ORON to orchestrate workflow executions. ORON nodes act as proxies to perform web service discovery, and web service executions on behalf of flowlets. These two schemes are discussed in the following sections.

6.5.2.1 Client-based Workflow Execution A flowlet performs workflow orchestration completely on the client node. It uses the ORON only to discover web services. The flowlet invokes each web service, and performs workflow state update directly. A flowlet begins a workflow process at the *initial marking* state. Starting from the *start* node defined in the workflow graph, the flowlet follows the outgoing arc to the next task node. Upon arrival at a task node, it executes the web service task defined on that node using the following steps:

- First, the flowlet discovers a collection of web services that match the service semantic from ORON. It uses a DHT-based routing and lookup strategy. It obtains a key by applying $hash()$ to the web service semantics. Then it sends a discovery query associated

with this key to an ORON node. Upon receiving the discovery query, the ORON node uses the DHT-based routing strategy to route the query to the nodes responsible for the web service semantics. The targeted ORON node then returns all web services registered with its registry to the flowlet. Note that there are several recent ontology-based matching methods for web services. They can be used to rank or filter out the matched web services. The detail of these methods can be found elsewhere [14, 69, 68, 72].

- Second, the flowlet selects the best web service with respect to the given constraints and goals. The flowlet applies the filter extracted from the given user preferences and the workflow constraints to the returned web services. In some cases, however, the filter can not be used directly on the returned web services. The flowlet may need to interact with these web services to find more information. This open issue is discussed in Section 6.7.
- Third, the flowlet invokes the chosen web service. It sends a web service invocation request encapsulated in a SOAP message to the web service.
- Forth, the flowlet collects the result returned from the web service to update the data and the workflow state. It updates all the condition(s) associated with the output places.
- Finally, the flowlet moves to the next enabled task node(s) in the workflow graph. Note that, by doing so, the flowlet may spawn or join processes based on the workflow patterns previously described in Section 6.1.2.1.
- The process continues until it reaches the *exit* node.

To further describe these procedures, the pseudo-code of this scheme is given below and illustrated in Figure 47.

Flowlet: Given a Personalized Workflow Profile $PWP = (G, D, W)$ where $W = (H, Dat), H = (T, C, E)$

```
1: Starts at the initial marking state
2: Executes the web service task t, associated with the enabled transition
3: for all service semantic s used to describe web service defined in task t do
4:   Sends PWP through DHT to discover web services associated with service
     semantic s at node responsible for hash(s)
5: end for
6: Collects the matched web service profiles those are returned from these nodes
7: Selects the best web services w.r.t the given goals (G)
8: Sends a soap message to execute the selected service
```

9: Collects the returned result
10: Updates Dat and all the conditions associated with the output places
11: Splits or joins the process to execute the next task(s)
12: Repeats step 2 until reaching the end node

ORON node: node n_i responsible for the semantic s

1: Retrieves the service profile indexed with the service semantic s,
2: Returns the matched web service profiles to the requesting flowlet

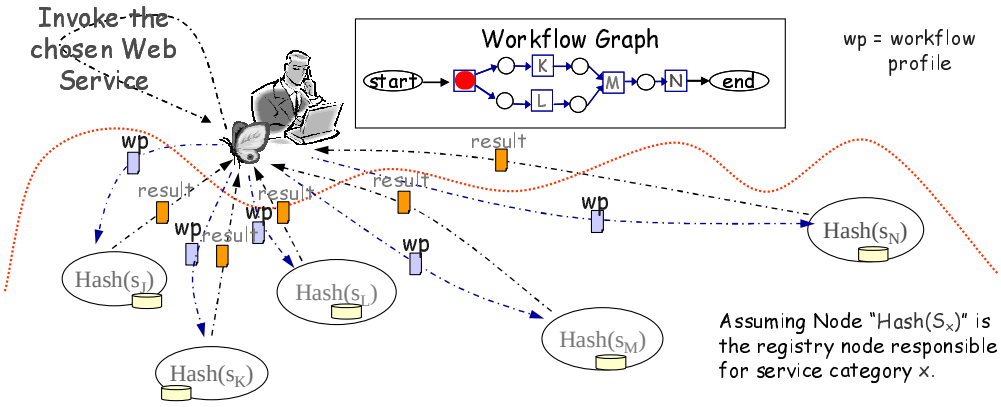

Figure 47: Client-based Workflow Execution Scheme.

The client-based scheme offers the flowlet full control of the workflow process. Each flowlet can have its own workflow engine to control a workflow process. At the same time, this scheme provides flexibility to a workflow implementation. Each flowlet can easily monitor and control a workflow process.

Despite all of the positive features of this scheme, there are drawbacks as well. This scheme incurs high communication overhead and forces the client to bear all the complexity of workflow execution. A flowlet requires redundant communication to and from the ORON. That is, for each web service task, a flowlet sends a web service discovery query to the ORON, and the targeted ORON node sends back all of the web services registered in its registry. If a large number of web services are registered in the ORON, the query causes a high volume of traffic.

In addition, each flowlet must implement a workflow engine to support its workflow execution. A flowlet must keep the workflow state up-to-date, and compute the next enabled tasks throughout the workflow process. It must also invoke the web service directly. On the other hand, the proxy-based workflow execution assigns most of the work to ORON. It is discussed in the section that follows.

6.5.2.2 Proxy-based Workflow Execution The proxy-based workflow execution is inspired by the notion of *routing slip* that defines the *routing schema*[97]. A *routing slip* is defined as "a simple sequence of roles or users who must review the documents in a certain sequence." A routing slip is normally sent with one or more documents to various workers for their actions. During an action, a worker may add, modify, or remove the attached documents. As a result, a sequence of workers to review these documents may be changed.

A distributed workflow process can be realized using the routing slip concept. A routing slip expressed in terms of a workflow graph can be distributed among parties who are engaged in a workflow. Upon receiving a workflow specification, each party then performs its task, updates the data and the state of the workflow process, and forwards the workflow process to the parties responsible for next tasks. The forwarding continues until the workflow process reach the end state.

The proxy-based scheme applies the routing slip idea to support the workflow execution. ORON nodes are considered as the workers for the workflow processes. A set of workers are dynamically assigned to a workflow process based on the semantics of its required web services.

Similar to the client-based scheme, the process starts at the *initial marking* state. The flowlet first computes the routing key(s) by applying the $hash()$ to the semantics of the first enabled task(s). It then sends a personalized workflow profile associated with the routing key to one of the ORON nodes. In turn, the ORON node routes the personalized workflow profile to the ORON node responsible for the service semantic.

Upon receiving a workflow specification, the ORON node then performs the web service task, updates the workflow state, and forwards it to the next ORON node responsible for the next tasks. In addition, to allow the flowlet to monitor the workflow process, the ORON node sends the flowlet an update of data and workflow state expressed in terms of conditions. The workflow process continues until an ORON node completes the end state of the workflow. Then, the data produced by the last node is sent back to the flowlet. The of this scheme is described below and illustrated in Figure 48.

Flowlet: Given a Personalized Workflow Profile $PWP = (G, D, W)$ where $W = (H, Dat), H = (T, C, E)$, and a set of condition(s) holding truth $P = \{start\}, where P \in C$.

1: Starts at the **initial marking** state
2: Sends PWP the ORON to perform workflow execution, starting at the enabled task
3: Setups an $alarm$ to time the workflow process
4: Waits for an $alarm\_timeout$ or $update$ event
5: if $update$ event then
6: Resets the $alarm$
7: Updates DAT, and P
8: if $end \in P$ then
9: Done
10: else
11: Repeats step 4
12: end if
13: end if
14: if $alarm\_timeout$ event then
15: Repeats step 2
16: end if

ORON node: node n_i responsible for the web service semantic s

---

1: Retrieves the service profile indexed with the service semantic s,
2: Selects the best web service w.r.t the given goals (G)
3: Sends a soap message to execute the selected service
4: Collects the returned result
5: Updates Dat and all the conditions associated with the output places define in PWP
6: Adds all the conditions holding truth as a result of the update to a condition set P
7: if condition at end place is $true$ then

```
 8:   Returns PWP to the flowlet
 9: else
10:   Returns DAT and P to the flowlet
11:   for all the web service task t that is enabled do
12:     Forwards PWP and P to the ORON node responsible for t's service semantic
13:   end for
14: end if
```

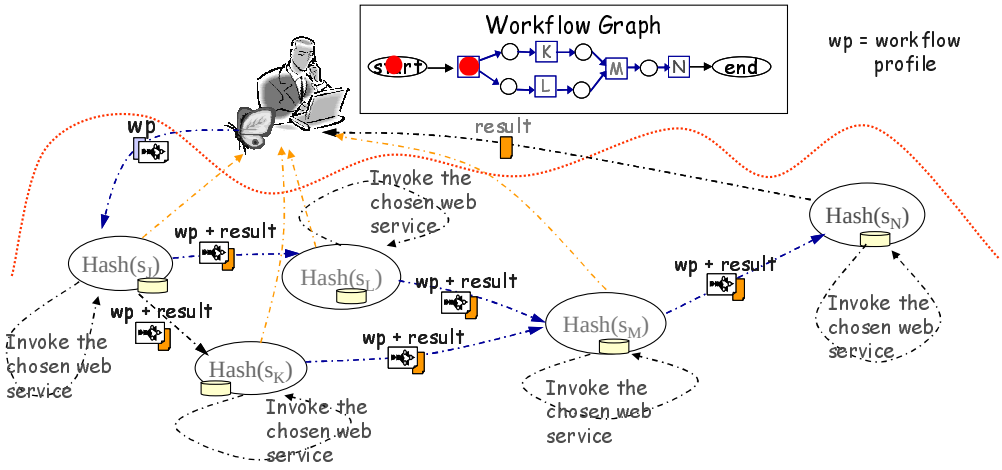

Figure 48: Proxy-based Workflow Execution Scheme

Compared to the client-based workflow execution scheme, this scheme provides a more efficient way to use resources in an ORON. The communication overhead is minimized by reducing the number of communications between the flowlet and the ORON. The web service discovery, and invocation costs are shared and distributed among ORON nodes.

Using this scheme, however, raises the issues related to robustness and security. These issues stem from the distribution of a workflow process. Since a workflow process is distributed among ORON nodes, it is more difficult for a flowlet to monitor and control the whole process. Each flowlet must perform extra work in monitoring and recovering the workflow when needed for each step of the process.

6.6 PERSONALIZED WEB SERVICE ARCHITECTURE (PWSA) COMPONENTS

This section describes the main components of a PWSA and how the components interact to support a web service workflow. These components are classified into three main categories: consumer components, service provider components, and ORON components. Two component interaction models based on the underlying workflow execution scheme are presented below.

6.6.1 Client-Based Component Interaction Model

The client-based component interaction model proposes a flowlet perform workflow execution on the client. A flowlet has a full control in workflow execution, and uses the ORON only to discover web services. The enabling components of the model include consumer components, provider components, and ORON components.

The consumer components include a set of workflow logics and flowlets. Each workflow logic is associated with a flowlet. In turn, the flowlet acts as workflow conductor to orchestrate web service tasks for the workflow logic. The flowlet then uses the workflow logic to guide interactions with the ORON and the service providers to achieve web service tasks.

The provider components include a web service and an advertising controller. The web service component interacts with service consumers to provide the service. The advertising controller interacts with ORON to advertise and register the web service in a CoSI.

The ORON components include a request dispatcher, a service discovery handler, a service advertising handler, a service registry, and a routing controller. They are responsible for web service discovery and advertising within a CoI.

The request dispatcher interacts with a consumer's flowlet and a provider's advertising controller. It identifies the type of request that arrives, and forwards it to the responsible handler. The consumer's discovery request is forwarded to the service discovery handler, and the provider's advertising request is forwarded to the service advertising handler. The service discovery handler is responsible for finding web services that match the request. It searches the local service registry, and returns the result to the flowlet.

At the same time, the service advertising handler is responsible for storing and indexing the advertising request. Additionally, it notifies the advertising controller to facilitate the service advertising process. The indexes and the data storage of a service registry are illustrated in Figure 49. The ontologies are used to index web service information. In the figure, the ontologies *OTA*, *eTom*, and *UNSPSC* are used. Note that only the web services associated with the ontological concepts that map to the ORON node are stored and indexed.

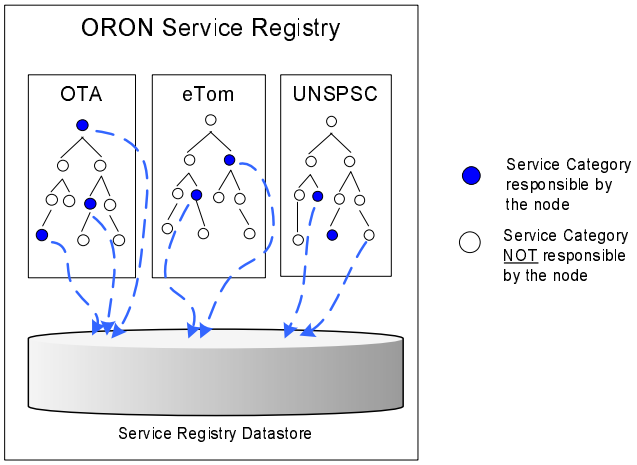

Figure 49: ORON Registry.

In order to route the request to the responsible ORON node, which is the responsibility of the routing controller, it interacts with a DHT component to find the neighbor that is closest to the routing key associated with the request. These components and their interaction partners are illustrated in Figure 50.

6.6.2 Proxy-based Component Interaction Model

Most of the components used to support the proxy-based scheme are similar to those used to support the client-based scheme. However, the differences between the client-based interaction model and the proxy-based interaction model include different ORON node components and different interactions to support a workflow execution.

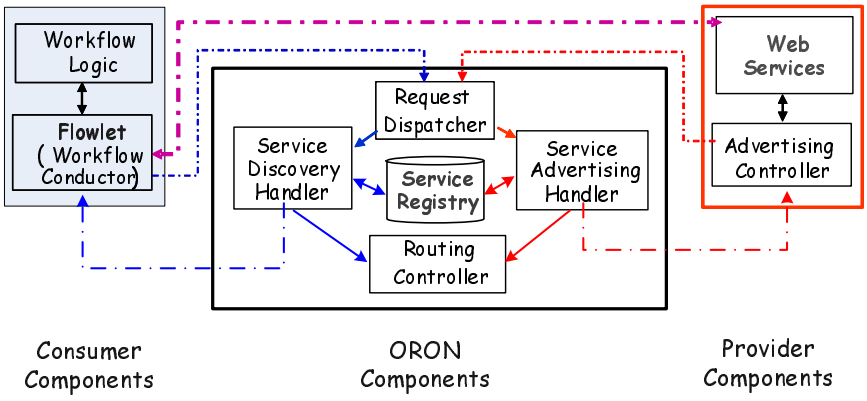

Figure 50: Component-Based Interaction Model of the Client-Based Scheme.

The Workflow coordinator replaces the service discovery handler. Its main function is not only to discover web services registered within the registry, but also to act on behalf of a flowlet to perform a web service task. This task includes invoking the chosen service, updating the workflow state, notifying the flowlet of the update, and interacting with the routing component to transfer the workflow process to the ORON node(s) responsible for the next enabled task(s).

The flowlet component at the client node reduces its role in workflow execution. The main function of the flowlet component is to monitor the workflow process, and to resubmit to ORON, in the case of an error, the workflow specifications and their up-to-date data and state to continue the workflow process. The components and their interaction partners are illustrated in Figure 51.

6.7 CONCLUSION

This chapter focused on the Personalized Web Service Architecture (PWSA) to support a user-based workflow execution. The architecture extends the Personalized Web architecture beyond dealing with static resources to dealing with dynamic resources provided by web

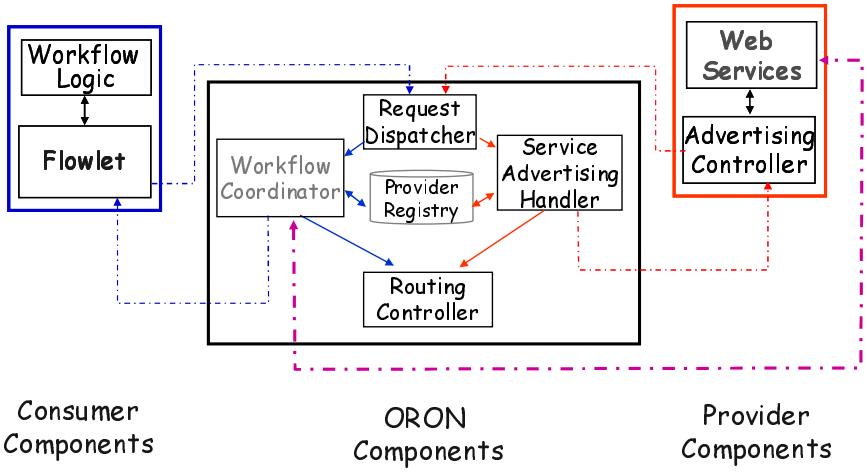

Figure 51: Component-Based Interaction Model of the Proxy-Based Scheme.

services. The aim was to demonstrate the capability of a Personalized Web architecture to support web service workflow applications. Scientists have predicted that web service and workflow will play a major role in the next generation Internet[70, 49, 11, 32]. By realizing the web service workflow applications, the PWSA demonstrates the full potential of Personalized Web architecture to support the resource discovery, access and sharing on the Internet.

PWSA employs two main components: a semantically-aware agent referred to as a flowlet, and an overlay infrastructure referred to as an Ontology-based Registry Overlay Network (ORON). A flowlet acts on behalf of a service consumer to perform workflow orchestration functions. The ORON acts as a rendezvous network between service consumers and service providers. It allows service consumers to meet service providers, who are in the same domain of interest, referred to as the community of similar interest(CoSI). Each web service in ORON is associated with a shared ontology, referred to as a Community Service Ontology(CSOnt). The uses of CSOnt allow the flowlet to efficiently and effectively automate web service discovery to support the workflow process.

This chapter has described the enabling framework and models, and discussed the main mechanisms to support workflow execution. All of this discussion has been in service of actualizing the PWSA described above. The main mechanisms discussed include web service advertising and registration, and web service workflow execution. The web service advertising and registration mechanism allows a service provider to advertise and register web services within their CoSI. By contrast, web service execution enables a flowlet to orchestrate workflow processes through ORON.

Two schemes to support the web service execution have been presented. The first scheme, referred to as the client-based scheme, proposes a flowlet perform workflow execution on the client. A flowlet uses ORON only to discover web services. The second scheme, referred to as the *proxy-based scheme*, proposes workflow execution be performed by OON nodes. This scheme is based on the concept of a *routing slip*. A routing slip specifies the sequence of tasks a set of workers need to perform on a set of documents. A workflow specification is viewed as a routing slip, and the OON nodes are considered as workers. A flowlet dispatches a workflow specification to the OON. A set of OON nodes then perform web service tasks, route the documents among them and update the state of the workflow process for the flowlet.

Comparing the two schemes, it becomes apparent that the client-based scheme provides the flowlet an easier way to control and monitor a workflow process. However, it incurs higher communications costs and it requires a workflow engine on the client. On the other hand, the proxy-based scheme offers better ways to minimize the communications costs and to share workflow engines through the OON nodes. Yet, using this scheme incurs higher costs in monitoring and controlling the workflow process, which is distributed among ORON nodes.

One can decide which scheme to use depending on how an ORON is formed and managed. If it is formed by the a highly collaborative network, whereby the security and robustness can be guaranteed at a high level, it is better to use the proxy-based scheme. However, if the OON is formed by several organizations that loosely coordinate, the client-based scheme should be used. This issue remains open for future research and exploration.

In conclusion, this chapter demonstrates that the concept of the Personalized Web can be applied to support a user-based web service workflow. A PWSA can be implemented over a network to support the existing web service infrastructure. This architecture allows for the efficient discovery of web services, while enabling consumers to acquire web services for a complex task in an automated manner. As a result, service consumers can realize the potential benefits of web service workflow.

7.0 CONCLUSION AND FUTURE RESEARCH DIRECTIONS

7.1 RESEARCH FINDINGS AND CONCLUSION

This dissertation addresses the Internet information overload problem which makes it difficult for users to locate resources of interests on the Internet. A novel approach based on a user-driven profile framework referred to as the *Personalized Web* is proposed. The Personalized Web, inspired by the Semantic Web, deals with the information overload problem by associating an explicit semantic ontology with resources to allow machines or agents to understand and interpret these resources autonomously. Furthermore, the Personalized Web approach uses personalized information to advertise, retrieve, access and share resources among users who share a similar interest referred to as a Community of Interest (CoI). The Personalized Web enables users to efficiently develop their own views of the Internet in the information space that reflects their interests closely. In addition, the Personalized Web approach employs agent-driven advertising and retrieval mechanisms and profile indexing over a P2P overlay architecture in order to allow a Personalized Web to evolve through time in a scalable and user-transparent manner.

Based on the proposed framework, this thesis aims to answer four main research questions. The first question examines whether or not it is possible to combine the P2P overlay network and agent-enabled Semantic Web technologies to enable a Personalized Web. If the answer to the first question is positive, the question that follows is: what mechanisms, protocols, and data structures must be in place to support personalized resource discovery, advertising, and access on the Internet? The third question then targets the performance objectives of the proposed mechanisms, protocols, and data structures. It explores how they can be provided in a scalable and robust manner. Finally, the fourth question assesses the future usefulness of the proposed framework and asks whether these mechanisms, protocols, and data structures can be extended to support next-generation web service workflow applications.

Chapter two answered the question of possibility of combining a P2P overlay network with an agent-enabled Semantic Web technology through the literature review of the P2P overlay architectures that support resource discovery, access, and sharing. A conclusion

arrived in Chapter two is that the P2P overlay architecture could be used to support the Personalized Web, yet the proposed architectures could not be used to support Personalized Web functionalities since they either lacked the agent support or did not include idiosyncrasies in resource advertising and discovery.

Chapter three followed up the issues discussed in Chapter two and addressed the second question regarding the mechanisms, protocols and data structures to support personalized resource discovery, access, and sharing. Chapter two proposed a Personalized Web architecture based on a DHT-based P2P overlay network. The proposed architecture is composed of three layers: the *agent layer* on the top, the *ontology overlay network (OON) layer* in the middle, and the *internet resource layer* on the bottom. The key components of this architecture are the *semlet* agent on the top layer and the *OON* on the middle layer. The semlets act on behalf of users to automatically advertise and retrieve resources of interest, dynamically form and manage CoIs, and persistently develop personal webs of resources that reflect the user's interest. Therefore, semlets facilitate users by continuously performing resource discovery, access, and sharing, as well as making these operations transparent to the users.

Each semlet is associated with a set of resources semantically described by a set of concepts defined by a community ontology referred to as *Comm-Ont* and a specific interest expressed in terms of *personalized semantic rules* and *similarity metrics*. The semlet uses the semantic concepts associated with the resources to advertise within its CoI, employs personalized semantic rules to retrieve resources, and utilizes similarity metrics to further filter out or sort resources based on their similarity to the user's resources. In this way, each semlet takes into consideration the individual user's idiosyncrasies when performing resource advertising and retrieval. Each developed Personalized Web contains only the resources that interest the user according to the user's resources. Note that the user's resources can be a single document describing topics of interest. The user's Personalized Web contains resources semantically described by these topics.

The OON behaves as the rendezvous point for semlets and as the proxy for resource advertising and retrieval. Its underlying DHT-based P2P overlay substrate and the ontology-based key mapping scheme provide efficient and robust mechanisms to index and locate data

on a large scale. It helps semlets to find and share objects of interests in an efficient and robust manner. At the same time, the OON enables semlets to discover other semlets who share similar resources and to form CoIs which, in turn, allow semlets to efficiently advertise and retrieve resources of interest among them. From this information, it can be seen that the OON enables semlets to perform their operations in an efficient, scalable, and robust manner.

Chapters four and five addressed the third question which examines whether or not the mechanisms, protocols, and data structures can be supported in a scalable and robust manner. Chapter four first discussed the mechanisms, protocols, and data structures necessary for the OON to support semlets in order to perform resource advertising and retrieval. The discussion focused on the OON due to the fact that the scalability and robustness of the Personalized Web architecture depend on the way the OON performs resource advertising and retrieval. The chapter presented three schemes namely the *aggressive* scheme, the *crawler-based* scheme, and the *the minimum-cover-rule* (MCR) scheme. These schemes were designed with different performance objectives which have a great impact on the scalability and robustness of Personalized Webs. The aggressive scheme stores and indexes multiple copies of data on multiple OON nodes in order to provide the data replication. On the other hand, the crawler-based scheme stores and indexes data only as necessary on the OON nodes in order to minimize the storage space on the OON. It provides no redundancy. However, in so doing it must refresh the advertising and retrieval query periodically to guarantee that the resources are discovered and advertised. Therefore, compared to the aggressive scheme, the crawler-based scheme uses more bandwidth. The MCR scheme leverages the benefits of both the aggressive and the crawler-based schemes. It stores and indexes data on the OON nodes at the minimum level where query refreshing is not required. Compared to the other two schemes, the MCR scheme uses less storage space than the aggressive scheme, while it consumes a similar amount or less bandwidth than the aggressive scheme does. Conceptually, the MCR is expected to perform best in a stable environment where data on the OON nodes is persistently available. However, it remains in doubt which one performs better in a churn environment where data on the OON nodes are inconsistently available since the OON nodes frequently leave and join the OON.

In order to evaluate the performances of the schemes presented in Chapter four, both a prototype implementation and a performance analysis framework were developed. They were described in Chapter five. This prototype integrated the Personalized Web components including the semlet and OON with the Bamboo-DHT. It implemented only a subset of the semlet and OON functions necessary to evaluate the proposed schemes. The experimental framework was composed of both emulation-based and simulation-based experiments. The experiments assessed the proposed schemes' performances in both stable and churn environments. The assessment was aimed at measuring the effectiveness and the efficiency of the schemes. The effectiveness metrics included the *precision* and the *recall*[1] of object advertising and retrieval. The efficiency metrics included the *storage space* and the *bandwidth* used for the schemes.

The results in a stable environment showed that the performances of the proposed schemes were as expected. Among these three schemes, the aggressive scheme and the MCR scheme were effective because they could obtain 100% precision and 100% recall, while the crawler-based scheme was less effective with only 100% precision. The reason for the ineffectiveness of the crawler-based scheme was that this scheme stored data only on a subset of the OON nodes responsible for the associated concepts and the refreshing of query could not recover data completely. However, the efficiency results showed that, while the crawler-based scheme was the most efficient in terms of storage space, the MCR scheme was the most efficient in terms of bandwidth consumption. Additionally, compared to the crawler-based scheme, the MCR scheme required a similar amount of storage space. As a result, the MCR scheme provided the most efficient and effective mechanisms for resource advertising and retrieval in a stable environment.

While the results in a stable environment demonstrate that the MCR scheme was the most efficient and effective, the result in a churn environment suggested that either the crawler-based scheme or the aggressive scheme might give a better performance. In this environment, nodes randomly join and leave the OON causing data to be inconsistently available to retrieve. Due to this inconsistency, the effectiveness of the MCR scheme, which

[1]The *precision* and *recall* concepts used to evaluate the performances of the proposed schemes are slightly different from those used in Information Retrieval research. See Section 5.3.

depended largely on the persistence of the data, was reduced considerably. By contrast, the performance of the aggressive and the crawler-based scheme deteriorated inperceptibly. The aggressive scheme coped with the churn of OON nodes by replicating data on multiple OON nodes, while the crawler-based scheme handled the churn by refreshing to achieve data recovery. Additionally, the refreshing interval played a crucial role in data recovery of the crawler-based scheme. In a high churn environment, the crawler-based scheme with a short refreshing interval provided the highest recall compared to those provided by the crawler-based scheme with longer refreshing intervals and those provided by the other two schemes.

The results in both stable and churn environments suggest that the OON that aims to support the Personalized Web in a churn environment must implement either a replication or a refreshing mechanism or both. The key parameters for replication and the refreshing mechanism must be carefully selected based on the churn rate of the OON nodes and on the required performance objectives. For example, in an environment where the churn rate is high, the OON must implement either replication with a high redundancy level or a refreshing mechanism with a small refreshing interval or both. Furthermore, if the performance objective is to minimize the bandwidth, the OON must maintain a high value for its refreshing interval. On the other hand, if the storage space is limited, the OON must maintain data replication at a low level. One may consider creating a mechanism to detect the churn of the OON nodes and adjust these key parameters to optimize the performance of the OON dynamically. This topic is left for exploration in the future.

Chapter six addresses the fourth question which explores whether Personalized Web architecture can be used to support the next generation of web service workflow applications. The workflow applications were selected to demonstrate case studies of the Personalized Web, because they were considered to be the next generation of Internet applications [70, 49]. Recently, many tools, standards, and frameworks have been developed to allow web services to be dynamically invoked, and possibly discovered through directory services. However, these web service applications suffer from the problem of information overload. On the one hand, service providers do not know where to advertise their services to reach their

target consumers. On the other hand, service consumers do not know where to discover web services that match their interests, or how to orchestrate these web services to work together to achieve their objectives. As a result, the service providers advertise their services to users who are not interested, and service consumers have to manually locate web services, wasting both time and energy. The dynamic nature of Web services makes it more of a challenge to support such applications. Automation tools are needed in order to dynamically find, put together, and orchestrate the available web services for consumers.

Chapter six describes how a Personalized Web framework could be extended to enable the automation of web service discovery, orchestration, and invocation for service customers. This chapter presented the extended architecture referred to as Personalized Web Service Architecture (PWSA), which was comprised of the Ontology Registry Overlay Network (ORON) and a flowlet agent. ORON is a special case of an OON designed to support web service indexing, advertising, and discovery. The flowlet, on the other hand, was developed from the semlet agent and its features were enhanced to support the automation of web service discovery and the orchestration referred to as web service workflow.

In order to enable an automation of a *web service workflow* using PWSA, two schemes designed with different constraints and performance objectives were presented. The first scheme was a centralized scheme where the flowlet performs each step of the web service workflow. The semlet employed the ORON to discover web services in each step. The second scheme, referred to as the decentralized "routing slip" scheme, distributes each step of the workflow to be performed at the ORON nodes. Using this scheme, the ORON nodes were responsible for web service orchestration and invocation. Both schemes have advantages and disadvantages. The centralized scheme is expected to provide a more robust and secure mechanism to perform web service workflow because it can detect and deal with failure and malicious behavior occurring in each step quickly. On the other hand, the decentralized approach is supposed to use bandwidth more efficiently since ORON nodes do not communicate back to the flowlet. Further analysis and exploration of key performance parameters of these schemes and experiments are needed.

Having addressed four main research questions in order to enable Personalized Web, several research directions could be suggested to explore the way to improve the performance of the proposed Personalized Web architecture. Moreover, the research directions could be advised to investigate the possible scenarios of using the Personalized Web architecture to support next generation Internet applications. The next section summarizes these suggested future research directions.

7.2 FUTURE RESEARCH DIRECTIONS

This thesis has developed a Personalized Web architecture which can contribute to many research areas including the Semantic Web, Ontology, Personalization, P2P Overlay Networking, Group Communication and Multicasting, Information Retrieval, Web Services Discovery, and Workflow. The following sections discuss future research directions involved with these research areas.

7.2.1 Semantic Web

Semantic Web research aims to make the web semantically understandable not only to humans but also to machines or software agents, such that the web can facilitate effective sharing of data between users on the Internet. This thesis has evolved in parallel to Semantic Web research. It focuses on the use of Semantic Web technology in order to enable users to advertise, discover, retrieve, and access Internet resources based on their interests within communities of interests. In other words, the knowledge sought and discovered in this study extends the possibilities of the Semantic Web, enabling it to enhance resource discovery and knowledge sharing on the Internet. This work described a Personalized Web architecture that allows users to associate resources with ontology-based profiles. These profiles make resources not only understandable but also activatable. Being understandable but activatable, they seek users who have an interest to advertise and form resource clusters specific to the users' interests. The Semantic Web can be realized quickly when users deploy the Personalized Web architecture.

Having proposed the Personalized Web architecture to realize Semantic Web, three main research questions should be addressed:

- How can web resources such as web pages with ontology-based profiles be associated to become a part of Personalized Webs?
- How can a suitable community ontology be found or developed to associate resources such that these resources are advertised and discovered among users who have similar interests?
- How can it be assured that the development of Personalized Webs are protected from the malicious behaviors such as denial of service (DoS) attacks?

To address the first question, there must be mechanisms and frameworks that facilitate users to create semantic profiles for the resources and to activate semlets to perform resource advertising and retrieval for them. Also, these mechanisms must be as user-transparent as possible and customizable to fit users' needs. A user could create an interest profile which specifies which topics are of interest and associate it with his or her web pages. An automated process then maps these topics with the ontologies that fit the profile and activates a semlet process to perform resource advertising and retrieval for the user's web pages based on the chosen ontologies. Upon the arrival of advertising, the user could then select the advertised resource to view if the user is interested, or delete it otherwise. Based on the user's interactions with the advertised resources, the semantic profiles for the resources could then be adjusted to reflect the user's interest. For example, the ontologies associated with the resources that the user views are promoted to be the main ontologies used for resource advertising and retrieval. On the other hand, the ontologies associated with resources that have been deleted are dropped from the user interest profile. To improve the sensitivity in automatic adjustment of ontologies, weights could be associated with these ontologies. These weights would then be dynamically adjusted to reflect the user activities such as searching, browsing, and viewing resources. Further exploration is needed in order to carefully design the mechanisms and frameworks that support such functionalities.

The second question examines how to find a suitable community ontology or develop one to associate with resources such that these resources are advertised and discovered among users who have similar interests. This work assumes that users who were in the same community of interest had the same perception of semantics and could access the same community ontologies. However, in the Internet environment this assumption cannot be made since users come from different places and diverse environments. Additionally, users do not know where to locate the ontologies. There must be mechanisms to allow users to discover such ontologies and allow them to select the ones that match their interests. Users also could define ontologies for their own communities. The system must allow users to publish their defined ontologies for others to view and use. One solution could employ directory services, which allow users to register, search, and retrieve ontologies. Further exploration is needed to find alternative mechanisms suitable for the open web environment.

The last question investigates the way to assure that the development of Personalized Webs is protected from the malicious behaviors such as denial of service (DoS) attacks. Due to the fact that the user's interests, expressed in terms of personalized semantic rules and semlet's addresses, are advertised and stored in the OON, users are vulnerable to semantically based DoS attacks. A malicious user may acquire these rules and semlets' addresses and attempt to continuously flood the semlets with bogus advertisements that match the semantic interest rules. In turn, the bandwidth of the hosts of these semlets are throttled and their processing resources are fully utilized to compute resource similarity. As a result, these semlet hosts are denied access to the OON or even any sites on the Internet. In order for this model to succeed, the Personalized Web architecture must provide mechanisms to prevent such attacks, and protect users' information within the OON. One solution could employ OON nodes to aggregate advertisements and periodically, instead of instantly, deliver these to the semlets. These OON nodes could also screen out advertisements which may be associated with unavailable resources or may come from suspicious hosts. Moreover, these OONs could encrypt or encode the personalized semantic rules and the semlets' addresses. It would then be difficult for malicious users to seek for the semantics to attack specific

semlet hosts. However, using such encryption or encoding requires processing resources on the OON nodes. Additional exploration is needed find the suitable levels of implementations of these operations.

7.2.2 Information Retrieval (IR)

Information Retrieval (IR) deals with the issues of representation, storage, organization of, and access to information items. IR provides techniques that can improve the OON indexing schemes and the agent's ability to find resources that match the user's interests. For instance, the stemming, stop-word elimination, and word sense disambiguation[2] techniques can be used to automatically analyze and produce accurate resource semantics associated with the resource profiles. The OON indexing scheme can apply the inverse document frequency (IDF) concept to index and store resources' and interest profiles on the OON nodes responsible only for the rarely used concepts. Using this application of the IDF, the OON can quickly filter out irrelevant profiles for queries associated with these concepts. As a result, the storage space required to store these profiles and query loads for the resources are effectively distributed among OON nodes. Future research in this area could explore how these techniques can help the development of Personalized Webs.

7.2.3 Web Personalization

Web personalization is a strategy that utilizes implicit or explicit user's information to leverage knowledge in a content delivery framework. As such, the information that specifically matches the users' interests can be presented to the users. A web personalization system works by first collecting information from users, then predicting interest from this information, and finally presenting matching items to the users. The simplest and most common approach used to acquire user information is the *explicit rating*, where users tell the system what they think about a piece of information. It is fairly well-understood and precise. However, having to stop to enter explicit ratings can alter normal patterns of browsing and reading. A less intrusive method is to use *implicit ratings*, where a rating is obtained by a

[2]The IR technique involves the association of a given word in a text with a definition or meaning (sense).

method other than obtaining it directly from the user. Web personalization normally uses both explicit and implicit ratings to acquire user information.

Various techniques have been proposed for the interest prediction from the user information. One of the well-known techniques is *collaborative filtering*, where automatic predictions (filtering) about the interests of a user are made by collecting taste information from many users (collaborating)[3]. The underlying assumption of this technique is that: Those who agreed in the past tend to agree again in the future. Using this assumption, a user who likes the same set of items as another user in the past will likely be interested in another item in which another user has an interest.

The techniques used for web personalization could be employed to improve the effectiveness of the Personalized Web. For instance, the combination of explicit and implicit ratings can be used to help derive precise personalized semantic rules that reflect users' interests. The OON can also employ collaborative filtering that suggests the objects of interest based not only on their explicit semantics but also on their history of selecting objects of interest within CoIs. Future research could explore how web personalization techniques can improve the effectiveness of the Personalized Web architecture and investigate the related implementation issues when these techniques are integrated with an OON. The issues to resolve include how to protect user information such as user preferences and ratings, and how to implement the distributed collaborative filtering normally employed in a centralized system where all user information is available.

7.2.4 P2P Overlay Networking

Recently, many P2P overlay networking research projects attempt to address the fundamental questions of deploying P2P overlay networking including how to provide fault-tolerance, security, and a bootstrap for P2P-based applications. To name a few, *replication* and *coding* mechanisms were proposed to deal with faults in the network. Secure routing including secure node ID assignment, secure routing table management, and secure message forwarding were

[3]The definition can be found in Wikipedia.com.

employed to improve the security of a structured-based P2P overlay network[16]. A public, structured-based P2P service, namely OpenHash, was created to provide a DHT-based P2P overlay infrastructure for applications[77].

Future research could investigate the mechanisms proposed in the P2P overlay networking research in order to improve the fault-tolerance and the security of the Personalized Web architecture. The underlying foundation of the Personalized Web architecture is a DHT-based P2P overlay which can implement such mechanisms. However, implementing these mechanisms will increase the complexity and redundancy of functions. For example, implementing a replication or coding scheme may be redundant due to the replication provided by the aggressive scheme. One could consider optimizing the benefits of using these mechanisms in the Personalized Web architecture.

Future research could also examine the deployment of an overlay network for the OON. This thesis presents an architecture that assumes the domains that participate in the OON have at least one node performing OON functions. Additionally, there exists a group of bootstrap nodes that allow nodes to join the OON. Realistically, client nodes may not want to perform OON functions, and a dedicated OON may be needed. A solution may use the public Open Hash service or a group of nodes provided by a private organization. Further experiments are needed to explore such a solution suitable for OONs.

7.2.5 Group or Multicast Communication

This thesis addresses the information overload problem of an architecture that allows users' semlets to discover other users' semlets who have a similar interest, such that they can form communities of interests (CoIs) which allow them to efficiently advertise and retrieve resources. In a small CoI, a semlet may advertise the user's resources directly to other semlets who are in the CoI. However, in a large CoI, direct, separate communication for each semlet may be prohibitively expensive and a group or multicast communication that allows one sender to send to multiple receivers in an efficient manner is needed.

During the last several decades, many algorithms have been proposed to improve efficiency in group or multicast communication. One common algorithm forms an overlay network that links all the receivers to the branching tree rooted from the sender [10, 48, 36]. In this way, a sender can send only a small number of messages to its subset of receivers and these receivers can continue on forwarding messages to other receivers and so on to the rest of its receivers. Some techniques use a structured-based P2P overlay network as the infrastructure to dynamically build a multicast-tree based on the overlay path that the request propagates [17, 74, 107]. Future research could explore and experiment with these algorithms to find schemes that semlets can use to communicate effectively and efficiently within their CoIs.

7.2.6 Web Service Workflow

The last future research direction suggested is to address the challenges of actualization of the work presented in Chapter seven in order to support web service workflow. Based on the proposed Personalized Web Service Architecture (PWSA) and two web service orchestration schemes, future research could explore the solutions to address the following issues:

- How can PWSA be used to automate workflows which involve sensitive information in a secure manner?
- How can reliability of flowlets be improved in performing workflow execution?
- How can the interoperability be provided especially in semantics among providers and consumers such that they can automate their workflows without human intervention?

In all three issues above, the issue of security in using PWSA to automate workflows is the most challenging to address. Using PWSA, flowlet agents need to carry user information and interact with web services to execute web service tasks. This information is exposed to the providers or any malicious users who falsely proclaim to be legitimate providers. In executing a web service task which requires user sensitive information, such as telephone numbers or credit card numbers, the architecture must provide a high level of protection or else refuse to deliver such information. At the same time, it must be flexible enough to allow flowlet agents to automate their given workflows. Flowlets could establish trust with service

providers through an electronic certificate authority. Encryption and encoding techniques also could be used to provide secure channels between flowlets and providers. However, in performing these operations extra processing resources are required which can be costly and can decrease the flowlets' ability to automate the workflows. Future research could provide mechanisms that make use of the security techniques while allowing flowlets to achieve their given workflows effectively.

Having addressed the first question, the second question involves finding a solution for reliability of flowlets in performing workflow execution. Since flowlets employ a group of nodes, virtually connected over the Internet, to perform web service discovery and orchestration for their workflows, they must be able to monitor and control their processes. Such processes may be distributed to execute on these nodes. In the case of failures, the flowlets can either retry alternative tasks or alternative web services, or roll back to their previous states if they choose to do so. If the flowlets roll back, they must be able to cancel previously executing processes. One could treat a given workflow which requires coordinated executions among the defined tasks as a *transactional workflow*[86]. The transactioinal properties, including atomicity, consistency, isolation, and durability for individual tasks or the entire workflow could then be used to enforce transactional executions among the tasks. Flowlets could log their activities interacting with the OON regarding their web service tasks, such that when a failure occurs, they can resume or cancel their workflow processes based on transaction properties of the tasks. OON nodes could also provide logging and trace services or even hold data or the state of selectively transactional workflows. Flowlets could then use these services to efficiently recover web service tasks and continue their executions. Future research could explore mechanisms and schemes that provide flowlets for these capabilities.

The last issue to solve pertaining of web service workflow relates to the question of how the PWSA provides interoperability especially in semantics among providers and consumers so that they can automate their workflows without human intervention. The PWSA must provide mechanisms that allow service providers to define service ontologies, publish them to the communities, and finally resolve conflicts in semantics between ontologies. The mechanisms may include using the ontology-based languages, automatically mapping mechanisms in service directories, and using P2P overlay infrastructure to exchange ontologies

[27, 57, 88, 81]. Exploration of how these mechanisms can help the PWSA to support flowlets and of the experiments to analyze their performances is needed.

7.2.7 Future Research Conclusion

In conclusion, future research is suggested to explore and examine the theories and practices of relevant research areas, including Semantic Web, Ontology, Information Retrieval, Web Personalization, Multicase Communication, P2P overlay networking, and web service workflow. These areas could improve the scalability and reliability of Personalized Web architecture and allow Personalized Webs to develop in a way that closely reflects users' interests. Technologies from the Semantic Web and ontologies could be directly used to describe resources and users' interests such that resources are well understood and users' interests can be precisely interpreted by semlet agents. Information retrieval techniques could increase the effectiveness of semlet advertising and retrieval. Web personalization mechanisms could allow semlets to interpret and predict users' interests in a user transparent manner. Multicast communication techniques could increase the efficiency and scalability of communications among members of CoIs. Finally, mechanisms and protocols of P2P overlay networking could improve the scalability and reliability of an OON.

In addition to the suggestions to improve the performance of Personalized Web architecture, future research could explore how the Personalized Web architecture can be expanded to realize next generation Semantic Web and workflow applications. Personalized Web architecture could be integrated into the core of the Internet to allow users to access an OON anytime and anywhere. The ubiquitous technologies expedited by increasingly improved wireless communications and sensor abilities could make resources or services provided by mobile, pervasive devices available to users. Through an OON, these resources could then be advertised to users who have interests and who are in the vicinity at the time these devices are providing services. These users can then use these resources in an ad-hoc and timely fashion. The Personalized Web could then be employed as a ubiquitous advertising and discovering point that allows users to find what they are looking for not only based on their interests but also based on times and places.

BIBLIOGRAPHY

[1] S. Ferrara A. Castano, S. Montanelli, and E. Pagani. Ontology-addressable contents in p2p networks. In *Preceedings of SemPGrid 03: 1st Workshop on Semantics in Peer-to-Peer and Grid Computing*, pages 55–67, Budapest, Hungary, May 2003.

[2] W.M.P. van der Aalst. The application of petri nets to workflow management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

[3] Shoukat Ali, Anthony A. Maciejewski, Howard Jay Siegel, and Jong0Kook Kim. Definition of a robustness metric for resource allocation. In *International Parallel and Distributed Processing Symposium (IPDPS'03)*, April 2003.

[4] OpenTravel Alliance. Xml schema for travel industry. Available online at http://www.opentravel.org/.

[5] Virgilio Almeida, Azer Bestavros, Mark Crovella, and Adriana de Oliveira. Characterizing reference locality in the www. In *The IEEE Conference on Parallel and Distributed Information Systems (PDIS)*, December 1996.

[6] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services: Concepts, Architectures, and Applications*. Springer-Verlag, 2004.

[7] David Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Proc. 18th ACM Symposium on Operating Systems Principles*, Banff, Canada, October 2001.

[8] Madhan Arumugam, Amit Sheth, and I. Budak Arpinar. Towards peer-to-peer semantic web: A distributed environment for sharing semantic knowledge on the web. In *The International World Wide Web Conference 2002 (WWW2002)*, Honolulu, Hawaii, USA, 2002.

[9] Magdalena Balazinska, Hari Balakrishnan, and David Karger. Ins/twine: A scalable peer-to-peer architecture for intentional resource discovery. In *Pervasive Computing*, August 2002.

[10] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *ACM SIGCOMM*, August 2002.

[11] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[12] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *Proc. of IEEE Conference on Computer Communications (INFOCOM'99)*, pages 126–134,, Mar 1999.

[13] Basic semantic web language. Online: http://infomesh.net/2001/07/bswl/, July 2001.

[14] Jorge Cardoso and Amit Sheth. Semantic e-workflow composition. *Journal of Intelligent Information Systems*, 21(3):191–225, November 2003.

[15] Fabio Casati, Ski Ilnicki, Li jie Jin, Vasudev Krishnamoorthy, and Ming-Chien Shan. Adaptive and dynamic service composition in eflow. *Proceedings of 12th International Conference on dvanced Information Systems Engineering (CAiSE)*, pages 13–31, June 2000.

[16] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Security for structured peer-to-peer overlay networks. *Fifth Symposium on Operating System Design and Implementation (OSDI 2002)*, December 2002.

[17] M. Castro, P. Druschel, A-M. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralised application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC)*, 2002.

[18] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, S.R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Hypersearching the web. *Scientific American*, June 1999.

[19] Edith Cohen, Amot Fiat, and Haim Kaplan. Associative search in peer to peer networks: Harnessing latent semantics. In *Preceedings IEEE Infocomm 2003*, April 2003.

[20] Arturo Crespo and Hector Garcia-Molina. Semantic overlay networks for p2p systems. Technical report, Computer Science Department, Stanford University, 2002.

[21] C. Cunha, A. Bestavros, and Mark Crovella. Characteristics of www client-based traces. Technical report, Boston University, July 1995.

[22] Frank Dabek, Jinyang Li, Emil Sit, Frans Kaashoek, Robert Morris, and Chuck Blake. Designing a dht for low latency and high throughput. *Proceedings of the USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, March 2004. The King data is available at http://pdos.csail.mit.edu/p2psim/kingdata/.

[23] Darpar markup language for services. Online: http://www.daml.org/services/.

[24] J. Dean and M. R. Henzinger. Finding related pages in the world wide. *WWW8 / Computer Networks*, 31:1467–1479, 1999.

[25] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic indexing. *American Society for Information Science*, 41:391–407, 1990.

[26] E. D. Demaine, A. Lopez-Ortiz, and J. I. Munro. Adpative set intersections, unions, and differences. In *The 11th Annual ACM-SIAM Symposium on Dicrete Algorithms (SODA 2000)*, January 2000.

[27] John Domingue, Dumitru Roman, and Michael Stollberg. Web service modeling ontology (wsmo) - an ontology for semantic web services. *The W3C Workshop on Frameworks for Semantics in Web Services*, June 2005.

[28] D.Tsoumakos and N. Roussopoulos. A comparison of peerto -peer search methods. In *The Sixth International Workshop on the Web and Databases*, June 2003.

[29] Open Financial Exchange. Ofx specification for financial industry. Available online at http://www.ofx.net.

[30] Georgios John Fakas and Bill Karakostas. A peer to peer (p2p) architecture for dynamic workflow management. In *Information and Software Technology*, volume 46, pages 423–431, 2004.

[31] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. The MIT Press, May 1998.

[32] D. Fensel and C. Bussler. The web service modeling framework wsmf. *Electronic Commerce Research and Applications*, 2002.

[33] Tim Finin and Joel Sachs. Will the semantic web change science? *Science Next Wave*, September 2004.

[34] A Keyword-Set Search System for Peer-to Peer Networks. Omprakash gnawali. Master's thesis, Massachusetts Institute of Technology, June 2002.

[35] TeleManagement Forum. Enhanced telecom operations map (etom). Available online at http://www.tmforum.org/.

[36] Paul Francis. Yoid: Extending the multicast internet architecture,. Technical report, ACIRI, 2000. Online: http://www.aciri.org/yoid/.

[37] The free network project. Online: http://freenet.sourceforge.net/.

[38] G. Furnas, TK. Landauer, L.M. Gomez, and S. Dumais. The vocabulary problem in human-system communication. *Communication of the ACM*, 30:964–971, 1987.

[39] L. Garces-Erice, P. Felber, E.W. Biersack, K.W. Ross, and G. Urvoy-Keller. Data indexing in dht peer-to-peer network. In *The 24th International Conference on Distributed Computing Systems (ICDCS-04)*, March 2004.

[40] Gnutella: Open source community. Online: http://gnutella.wego.com/, 2001.

[41] Groove networks. Online: http://www.groove.net.

[42] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. In Nicola Guarino and Roberto Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, March 1993.

[43] K. P. Gummadi, R. Gummadi, S. D. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of dht routing geometry on resilience and proximity. In *ACM SIGCOMM*, 2003.

[44] P. Haase, J. Broekstra, M.Ehrig, M. Menken, P.Mika, M. Plechawski, P. Pyszlak, B. Schnizler, R. Siebes, S. Staab, and C. Tempich. Bibster - a semantics-based bibliographic peer-to-peer system. In *The International Semantic Web Conference (ISWC2004)*, November 2004.

[45] Martin Hepp. A methodology for deriving owl ontologies from products and services categorization standards. In *The 13th European Conference on Information Systems (ECIS'05)*, May 2005.

[46] T. Hofmann. Probabilistic latent semantic indexing. *Research and Development in Information Retrieval*, pages 50–57, 1999.

[47] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42:177–196, 2001.

[48] Yang hua Chu, Sanjay G. Rao, Srinivasan Seshan, and Hui Zhang. Enabling conferencing applications on the internet using an overlay multicast architecture. *ACM SigCOMM*, August 2001.

[49] Gretchen Hyman. Ibm to push grid-based web services. Online url:*www.internetnews.biz/ent-news/article.php/1015631*, April 2002.

[50] Sam Joseph. Neurogrid: Semantically routing queries in peer-to-peer networks. In *The International Workshop on Peer-to-Peer Computing (co-located with Networking 2002)*,, Pisa, Italy, May 2002.

[51] Vana Kalogeraki, Dimitrios Gunopulos, and D. Zeinalipour-Yazti. A local search mechanism for peer-to-peer networks. In *the eleventh international conference on Information and knowledge management*, pages 300–307, November 2002.

[52] Kazaa file sharing network. Online: http://www.kazaa.com, 2002.

[53] Bo Leuf. *Peer to Peer*. Addison-Wesley, June 1992.

[54] J. Li, B. T. Loo, J. Hellerstein, F. Kaashoek, D.R. Karger, and R. Morris. On the feasibility of peer-to-peer web indexing and search. In 2^{nd} *International Workshop on Peer-to-Peer Systems (IPTPS'03)*, June 2003.

[55] Alexander Loser. Towards taxonomy-based routing in p2p networks. In *Workshop on Semantics in Peer-to-Peer and Grid Computing*, pages 35–50, Newyork, USA, May 2004.

[56] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *The 16th international conference on Supercomputing*, pages 84–95, July 2002.

[57] David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, and Katia Sycara. Owl-s: Semantic markup for web services. Online URL: http://www.w3.org/Submission/OWL-S/, November 2004.

[58] Drew McDermott. Estimated-regression planning for interactions with web services. *Proceedings of 6th International Conference on AI Planning and Scheduling*, 2002.

[59] Deborah L. McGuinness, Richard Fikes, James Hendler, and Lynn Andrea Stein. Daml+oil: An ontology language for the semantic web. *IEEE Intelligent Systems*, 17(5):72–80, October 2002.

[60] Sheila A. McIlraith and David L. Martin. Bringing semantics to web services. *IEEE Intelligent Systems*, 18:90–93, 2003.

[61] B. Medjahed, A. Bouguettaya, and A. Elmagarmid. Composing web services on the semantic web. *The International Journal on Very Large Data Bases (VLDB)*, November 2003.

[62] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-peer computing (technical report hpl-2002-57). Technical report, HP Lab, Mar 2002.

[63] M. Mitzenmacher. Compressed bloom filters. *20th ACM Symposium on Principles of Distribututed Computing*, August 2001.

[64] Morpheus file sharing system. Online: http://www.musiccity.com/, 2002.

[65] Tadao Murata. Petri nets: Properties, analysis and applications. *The Proceedings of the IEEE*, 77(4):541–580, April 1989.

[66] Kiyohide Nakauchi, Hiroyuki Morikawa, and Tomonori Aoyama. Semantic peer-to-peer search using query expansion. In *The Second Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPGrid)*, May 2004.

[67] Srini Narayanan and Sheila A. McIlraith. Simulation, verification and automated composition of web services. *Proceedings of the 11th International World Wide Web*, pages 77–88, May 2002.

[68] Swapna Oundhakar, Kunal Verma, Kaarthik Sivashanmugam, Amit Sheth, and John Miller. Discovery of web services in a multi-ontology and federated registry environment. *International Journal of Web Services Research*, 2(3):1–32, July 2005.

[69] Massimo Paolucci, Katia Sycara, Takuya Nishimura, and Naveen Srinivasan. Using daml-s for p2p discovery. In *International Conference on Web Services*, 2003.

[70] Paulo F. Pires, Mario Benevides, and Marta Mattoso. Building reliable web services compositions. *Proc. Intl. Conf. on Web Databases and Web Services*, pages 59–72, 2002.

[71] C. Greg Plaxton, Rajmohan Rajaraman, and Andrea W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 311–320, 1997.

[72] J. Rao and X. Su. Toward the composition of semantic web services. *Proceedings of the Second International Workshop on Grid and Cooperative Computing*, December 2003.

[73] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *ACM SIGCOMM*, 2001.

[74] S. Ratnasamy, M. Handley, R. Karp, and S Shenker. Application-level multicast using content-addressable networsk. In *NGC'01*, November 2001.

[75] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *The 14th International Joint Conference on Artificial Intelligence*, pages 448–453, Montreal, 1995.

[76] Patrick Reynolds and Amin Vahdat. Efficient peer-to-peer keyword search. In *Middleware 2003*, 2003.

[77] Sean Rhea, Brighten Godfrey, Brad Karp, John Kubiatowicz, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, , and Harlan Yu. Opendht: A public dht service and its uses. In *ACM SIGCOMM'05*, August 2005.

[78] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.

[79] Kedar Samant and Siddhartha Bhattacharyya. Topology, search, and fault tolerance in unstructured p2p networks. In *The 37th Hawaii International Conference on System Sciences (HICSS'04)*, January 2004.

[80] C. Sangpachatanaruk and T. Znati. A p2p overlay architecture for personalized resource discovery, access, and sharing over the internet. In *CCNC'05*, January 2005.

[81] Mario Schlosser, Michael Sintek, Stefan Decker, and Wolfgang Nejdl. A scalable and ontology-based p2p infrastructure for semantic web services. In *Second International Conference on Peer-to-Peer Computing (P2P'02)*, pages 104–111, September 2002.

[82] Cristina Schmidt and Manish Parashar. Flexible information discovery in decentralized distributed systems. In *12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03)*, pages 226–236, Seattle, Washington, June 2003.

[83] Cristina Schmidt and Manish Parashar. A peer-to-peer approach to web service discovery. *World Wide Web Journal*, 7, June 2004.

[84] H. Schuster, D. Georgakopoulos, A. Cichocki, and D. Baker. Modeling and composing service-based and reference process-based multi-enterprise processes. *Proceedings of the 12th International Conference on Advanced Information Systems Engineering (CAiSE)*, June 2000.

[85] Seti project. Online: http://setiathome.ssl.berkeley.edu/.

[86] Amit Sheth and Marek Rusinkiewicz. On transactional workflows. *Special Issue on Workflow and Extended Transaction Systems (Data Engineering Bulletin)*, 16, June 1993.

[87] Clay Shirky. What is p2p... and what isn't. *The O'Reilly Network*, November 2000.

[88] Naveen Srinivasan, Massimo Paolucci, and Katia Sycara. Adding owl-s to uddi, implementation and throughput. In *First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, September 2004.

[89] Kunwadee Sripanidkulchai, Bruce Maggs, and Hui Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *Preceedings IEEE InfoComm 2003*, 2003.

[90] I. Stonica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishman. Chord: A scalable peer-to-peer lookup service for internet applications. *IEEE/ACM Transactions on Networking*, 2001.

[91] Chunqiang Tang, Zhichen Xu, and Sandhya Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *ACM SIGCOMM 2003*, pages 175–186, August 2003.

[92] Chunqiang Tang, Zhichen Xu, and Mallik Mahalingam. psearch: Information retrieval in structured overlays. In *First Workshop on Hot Topics in Networks (HotNets-I)*, New Jersey, USA, January 2003.

[93] Doug Tidwell. Web services: the web's next revolution. Online: http://www-128.ibm.com/developerworks/edu/ws-dw-wsbasics-i.html, November 2000. Accessed January 2006.

[94] Doug Tidwell. Web services: the web's next revolution, ibm tutorial. Online http://www-128.ibm.com/developerworks/edu/ws-dw-wsbasics-i.html, November 2000. Last visited January 2000.

[95] Dimitrios Tsoumakos and Nick Roussopoulos. Adaptive probabilistic search for peer-to-peer networks. In *Third International Conference on Peer-to-Peer Computing (P2P'03)*, pages 102–110, September 2003.

[96] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker. Scalability and accuracy in a largescale network emulator. In *OSDI'02*, December 2002.

[97] W.M.P. van der Aalst and A. Kumar. Xml based schema definition for support of inter-orgnaizational workflow. *Information Systems Research*, 14:23–46, 2003.

[98] Kunal Verma, Kaarthik Sivasshanmugam, Amit Sheth, and Abhijit Patil. Meteor-s wsdi: A scalable p2p infrastructure of registries for semantic publication and discovery of web services. In *Journal of Information Technology and Management*, volume 6, pages 17–39, January 2005.

[99] Margie Virdell. Business processes and workflow in the web services world. Online http://www-128.ibm.com/developerworks/webservices/library/ws-work.html, Jan 2003. Accessed February 28, 2006.

[100] W3C. Xml path language (xpath) 1.0. Online: http://www.w3.org/TR/xpath/, November 1999.

[101] Web ontology language (webont). Online http://www.w3.org/2001/sw/WebOnt/, 2001.

[102] Jared Winick and Sugih Jamin. Inet-3.0: Internet topology generator, 2001.

[103] I. H. Witten, A. Moffat, and T.C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, May 1999.

[104] D. Wu, E. Sirin, J. Hendler, D. Nau, and B. Parsia. Automatic web services composition using SHOP2. *Workshop on Planning for Web Services*, June 2003.

[105] Jun Yan and Yun Yang. Swindew – a p2p-based decentralized workflow management system. In *IEEE Transactions on Systems – Man and Cybernetics*, 2005.

[106] B. Y. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. *UCB Tech. Report (UCB/CSD-01-1141)*, April 2001.

[107] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proc. of Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, June 2001.