

iShare - Open Internet Sharing Built on Peer-to-Peer and Web

Xiaojuan Ren and Rudolf Eigenmann

School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN, 47907, USA
{xren,eigenman}@purdue.edu

Abstract. This paper presents design concepts and implementation overview of an Internet-sharing system, iShare. iShare supports end users as well as providers of Internet resources in disseminating, accessing and using these resources, in a way that allows open participation. A fully decentralized organization allows providers to simply post their resources on any web page, imposing no restrictions on resources attributes, administrative rules, and access protocols. Underneath its user surface it employs peer-to-peer information dissemination, advanced resource matching, open migration, and automatic service portal mechanisms. In addition to the qualitative comparison with related work, we evaluate the system in terms its efficiency of resource discovery and job execution.

1 Introduction

Internet-sharing systems harness the rapidly growing, worldwide resources of computer, network, and information systems. Advanced sharing technology bears tremendous potential in creating synergy among the users – both providers and end users – of machine platforms, networks, computer applications, software services, and information. We envision that future end users will be able to quickly learn about the availability of world-wide resources . They will be able and to employ them as if they were located nearby, without download, installation, or maintenance effort. Becoming familiar with new resources will be easy, despite large varieties in language, culture, and education. Providers of resources will be able to easily offer their new software, hardware, or data to the community. In doing so, they will be able to define their own rules and create business or open source models, akin to today’s economic principles.

In this paper we present design concepts and implementation overview of an Internet-sharing system, iShare, which serves as a research platform in pursuit of this vision. iShare facilitates sharing of three types of resources: programs (software tools, computational applications), machines (computers, devices), and data (documents, data bases). While much technology exists today for the sharing of data, the focus of iShare’s initial thrust is on technology for sharing executable programs (a.k.a. *services* – we will use the two terms interchangeably) and their underlying compute platforms.

iShare extends concepts of the PUNCH [9, 8] network computing system, which became operational in 1995 at Purdue University and has since served a large user community (over 3000 users in 35 countries) in computational nanotechnology, computer architecture and parallel programming. Both iShare and PUNCH have a strong end-user orientation. They 1) provide the means to disseminate, access and use networked resources and 2) they populate the infrastructure with services for the community. This orientation distinguishes the systems from related work in areas such as Grid computing and Web services. While these areas are pursuing goals similar to the original PUNCH project, their current focus is on developing the underlying technology, programming support, and standards for exploiting computational resources. iShare was motivated by an observed limitation of PUNCH, which it shares with many related efforts: populating the infrastructure with additional resources is limited by eligibility requirements and administrative procedures. For example, machines may only be added if they install certain server software or adopt certain administrative procedures; and programs may only be added after they become “grid-enabled”. Furthermore, the new resources may need to be registered in or approved by a central organization before they become visible to the community. Removing these barriers would enable Internet resources to be shared in a truly scalable manner.

iShare addresses these issues by taking an *open publication* approach, which imposes no restrictions on participating resources. A web-posting mechanism publishes new resources and an underlying, fully distributed peer-to-peer mechanism supports resource discovery. Powerful resource matching mechanisms serve to map program resources onto fitting platforms. While iShare builds on existing standards and middleware technology, *protocol plug-ins* allow new protocols and standards to be added, satisfying the needs of resources posted in the future.

The remainder of this paper is organized as follows. Section 2 describes iShare’s key design concepts. Section 3 gives an overview of the realization of these concepts. Section 4 provides evaluation results. We discuss related work in Section 5, followed by conclusions in Section 6.

2 Design Concepts

An Internet-sharing system can be characterized in terms of 1) its user model, 2) the semantics and type of the resources it can support, 3) the information disseminated about resources and activities at remote sites, and 4) the mechanisms that support the use of these resources at remote sites. This section elaborates on iShare’s distinguishing design concepts in these four areas, as shown in Figure 1.

2.1 User Model

Providers are important users: Internet-sharing systems provide their user communities with facilities to discover and make use of resources created and maintained at remote sites. In addition to its *end users*, iShare considers *providers* of resources as an important user class. Resource developers take advantage of

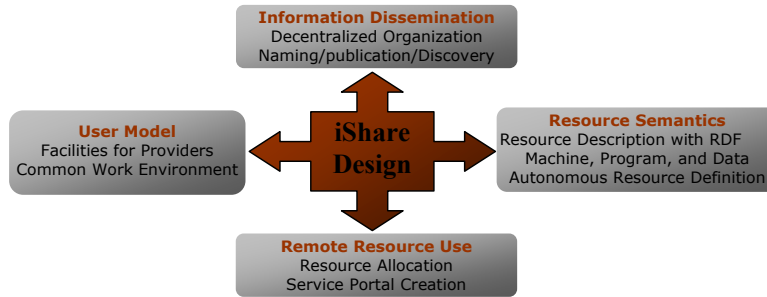


Fig. 1. Distinguishing design concepts of the iShare Internet-sharing system.

the ease with which resources can be made available to end users. They must be allowed to offer resources under their own rules, administrative procedures, and access protocols.

Access through common work environments: Web portals provide convenient user interfaces to many Internet-sharing systems. They are accessible from any Internet-connected platform and support hyperlink-based, graphical interactions. However, advanced users often prefer to work within their common environment, such as a Unix shell or Windows desktop. In general, the user model will be integrated into different local computing environment. Furthermore, new Internet appliances may want to provide their own user interfaces to the Internet-sharing functionality. Our design provides an API of the user functionality, on top of which interfaces for Web, Unix, Windows, and external devices can be built.

2.2 Resource Semantics and Types

RDF - semantic-oriented resource description language: An important issue in Internet-sharing systems is the clear description of resources. Resource providers must define the semantics about what is shared, who is allowed to share, and how sharing occurs. On the other hand, semantics interchangeability asks for standard definitions of resource characteristics. iShare adopts RDF as the resource description language for consistent encoding of resource attributes. RDF aims to specify semantics for data in a standardized interoperable manner [1]. The language provides a rich data model for describing objects.

In iShare, resources are described in a format that builds on W3C's RDF (resource description format) [2]. This XML-based format is described in Section 3.2.

Program, machine, and data resources: iShare supports three types of resources at an equal level: software services (programs), service platforms (machines), and data. An important program attribute is that of a *pinned* or *roaming* service. Pinned services are applications running only on a specific platform, whereas

roaming applications can execute on any matching platform. Machines are published as available hosts for roaming services. They might be individual machines or locally managed sub-systems (e.g., a network of workstations managed by Condor [12]).

Autonomy in defining resource attributes: An important design concept is that resources can define their own rules of usage. Thus, a machine may offer services under its own administrative procedures and access protocols. iShare will find matching programs and platforms. Protocols that are not available initially may be plugged in, making the system incrementally more powerful.

2.3 Information Dissemination

Decentralized organization: The design of an Internet-sharing system is implicitly tied to the nature of the information required to support the network-accessible computing. Many types of information may need to be disseminated in an Internet-sharing system. Examples are resource descriptions, job execution profiles, resource-usage information, and user profiles. The information is incremental and dynamic in the sense that it is created from items available at different points in time. Also, the information is often defined by independent users at different locations. The incremental, dynamic, and distributed nature of the information makes it infeasible to collect and maintain it at a centralized location in a scalable manner. Thus, a decentralized scheme is essential for reliable and efficient information management. In iShare, the decentralized scheme is realized as a P2P-Web integrated structure, which is described below.

Information naming, publication and discovery: Information dissemination typically involves 1) information naming schemes, 2) techniques for storing the information, and 3) mechanisms for entering and querying the information in the repository. In iShare, information is organized in a hierarchical name space and stored in a distributed repository, built on a structured P2P network. The P2P system is able to scale without the need for powerful servers by distributing the main costs of information sharing - space and bandwidth - across the peers in the network. iShare resources are announced by simple web posting. The resource characteristics, including the URL of the posted web page, are inserted into the P2P network. This provides a way to gather these postings via resource discovery built on efficient P2P message routing.

2.4 Remote Resource Use

Mechanisms for the remote use of resources are at the core of most Internet-sharing systems. They include the functionality for matching discovered software services with execution platforms, employing suitable protocols for remote job execution, and connecting user input/output. iShare builds on a large number of contributions in this area. Two features distinguish iShare's remote resource use: support for roaming services and automatic generation of service portals.

Resource matching for roaming services: Roaming services are programs that pick the most suitable platform for every invocation. The challenge faced by iShare is that, in contrast to related approaches, target platforms are not guaranteed to be able to run the desired program. Our concepts include advanced resource matching, which also considers past performance, and *open migration* of the service to the matching platform. Advanced matching needs to consider the option that services whose program source code was included in the publication may be recompiled and assigned to a different platform. Decision making for such matching in heterogeneous environments involves replication and caching strategies for service binaries and performance prediction techniques that consider possible recompilation. Open migration installs a service on a potentially unreliable platform. This involves monitoring, safeguarding mechanisms, and possibly relocating the service.

Automatically created service portals: The service portal establishes the user interface to a remote program. The challenges are to create this interface automatically from the service description given by the provider and to generate it in the user's preferred software environment. Service portals may be batch-oriented or fully interactive.

3 The iShare Architecture

In this section we describe the realization of the design concepts and their integration into the overall iShare system. The system architecture is shown in Figure 2(b). For space reasons, several important iShare components were left out of this paper – most notably iShare's authentication and trust mechanism (building on state-of-the-art techniques) as well as facilities for exchanging experience and bridging across diverse user communities. An extended version of this paper describes these components [7].

3.1 User Layer: Realizing the User Model

Supporting a *Resource Provider* User Class The basic iShare functionality for resource providers supports 1) creating resource descriptors, 2) publishing and removing resources, and 3) optionally starting and configuring advanced iShare services (e.g., job monitoring, iShare built-in job submission protocols). Publication of a new resource is fully autonomous and does not require user interaction with any iShare “administrator”. To end users, the published resources appear organized into discipline-specific *Cyberlabs*. Providers define (as a resource attribute) the Cyberlabs in which they wish to “place” the new resource. Currently, iShare contains five labs that include software tools from chemistry, biology, computer security, compiler and nanotechnology. As end users discover newly published resources they can place them in different Cyberlabs, if preferred. The core functionality for end users is to run the discovered, remote programs, using local files and displays as input/output.

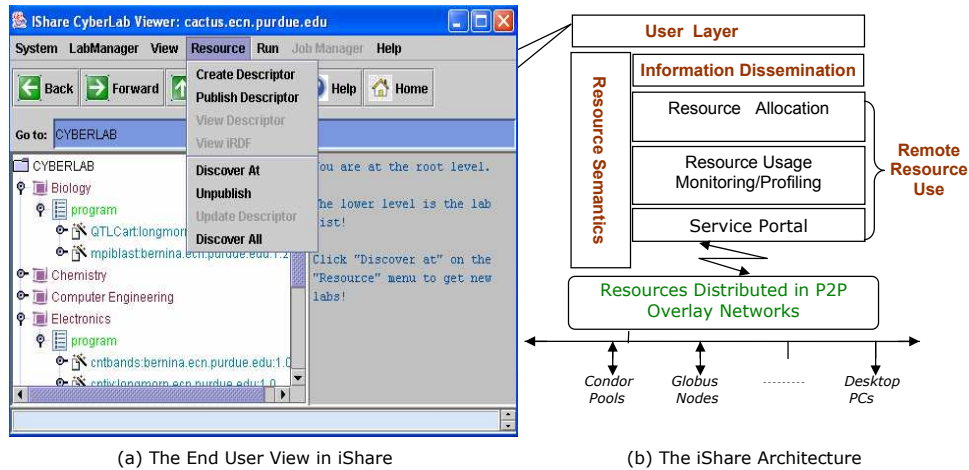


Fig. 2. The End User View and the Structure of iShare.

Providing iShare Functionality Within Common Work Environments

The user functionality is accessible via an API, which allows different user interfaces to be built. Our initial design includes a Windows-based interface, as shown in Figure 2(a) and we are exploring iShare Unix shell functionality. For example, in a Unix environment, the executable *path* (environment variable) is extended to include iShare resources. Thus, when executing a Unix command, a program is first searched locally and then among the iShare resources. Remote programs are run via iShare’s resource allocation and remote execution functionality. User interfaces will also include web portals, as modeled by the predecessor project, PUNCH.

3.2 Support for Resource Semantics and Types

Resource Description with RDF The resource semantic component provides tools for creating and compiling resource descriptors encoded in RDF. RDF descriptors include *metadata* describing resource attributes. Typical resource metadata for software services include location, connection protocol, execution syntax, pre (post)-process operations and platform requirement. Machine resources are characterized by their contact address/port, access protocols and capabilities. Resources in iShare are hierarchically categorized into *Cyberlabs*, as shown in Figure 3. The resources in one lab are semantically related in their functionalities. Resources are described by metadata, which form a tree representing the hierarchical name space. Instead of maintaining central directory service for the tree, iShare distributes the hierarchical name space to a P2P network, which supports the publication and discovery process described in Section 3.3.

Autonomy in Defining Resource Attributes To ensure semantic interchangeability in resource descriptors, it is essential to define the standardized

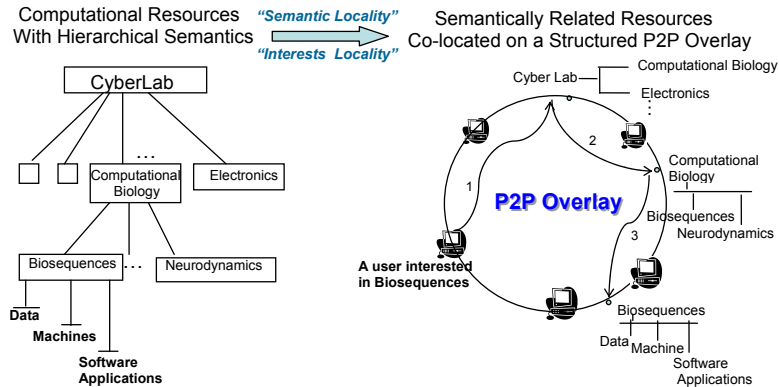


Fig. 3. Resources are organized in a hierarchical tree, which is mapped to a P2P overlay. The big circle represents a P2P overlay, with arrows indicating P2P routing messages to discover resources.

metadata for resource description. On the other hand, To support the autonomy in defining local access rules, the standardized metadata definition must be extensible enough to support plug-in protocols. Here, the autonomy means that if an access is permitted in an individual system, it must also be permitted under the cross-domain environment in iShare. This poses challenges in describing resource access protocols. The metadata definition needs to include enough details so that iShare’s computing environment could adapt to the plug-in protocols without user intervention. In our initial design, we formalized the metadata to include the common attributes of access protocols in widely used local management systems such as Condor [12] and PBS [13]. The metadata includes recompiling/relinking operations, submission syntax and user-commands. Here, we use Condor as an example to describe these metadata. Submitting a job to Condor involves recompiling/relinking to utilize checkpointing and process migration. And Condor’s job manager only accepts “submission files” compiled with predefined syntax, including run-specific, site-specific and application-specific information. Remote job runs are initiated/monitored via specific commands: `condor_submit` and `condor_rm`. All this information must be described in the resource access protocols. The information is then used for creating service portals, which hide the complicated details from end users.

3.3 Information Dissemination Mechanisms

Decentralized Organization Built on P2P and Web A key idea of iShare’s decentralized structure is that a provider of resources can post their availability on any web page. Resource metadata is derived from these postings and inserted into a P2P network. While the World Wide Web affords unprecedented access to the resource descriptors, metadata distributed in the P2P network improves discovery of and access to resources.

The P2P overlay in iShare consists of all participants. An iShare node could be a host serving a published software service, a host published as an accessible machine resource, or the workstation from which an end user accesses iShare. A participant joins the overlay network as a new peer node (super-peer) or by connecting to an existing peer node (p-client). A super-peer is a node in a peer-to-peer network that operates both as a server to a set of clients [5], and as equal in an overlay of super-peers. A p-client connects to a super-peer node and interacts with the overlay network through the super-peer. A new node with higher capabilities, e.g., bandwidth and processing power, tends to act as a super-peer. Nodes with lower capabilities join as a p-client by establishing a TCP connection with the closest super-peer node. By exploiting the heterogeneity among participating nodes, the super-peer structure avoids the potential bottlenecks caused by nodes with limited capabilities.

To publish a new resource, the publisher specifies the publication URL through the user layer configuration. Resource descriptors are posted on this URL manually or automatically through iShare's resource publishing tool. The involved web servers serve as a repository to resource description documents and are managed by the individual resource publishers. The P2P network in iShare consists of all participant nodes. An iShare node could be a host serving a published software service, a host published as an accessible machine resource, or the workstation from which an end user accesses iShare. A participant joins the P2P network as a new peer node (super-peer) or by connecting to an existing peer node (p-client). A new node with higher capabilities, e.g., bandwidth and processing power, tends to act as a super-peer. Nodes with lower capabilities join as a p-client by establishing a TCP connection with the closest super-peer node.

Another use of the web in iShare is that, publishers of roaming services put the installable program on a web server. Once the service platform(s) are located, the *open migration* component downloads the service package from the web server and checks it. Figure 4 shows the combination of HTTP and P2P communications when a roaming service is requested.

Information Naming, Publication and Discovery Information disseminated in iShare includes resource descriptions, job execution profiles, resource-usage information, and user profiles. Each piece of information is linked to a specific resource. Thus it could be also mapped to the hierarchical structure described in Section 2.3. The whole hierarchical name space is mapped to the underlying P2P network. An item in the hierarchical space is mapped to a peer node by the hashing value of the item's prefix path. A child's path name, instead of its mapped node's id, is kept in the parent's repository. Therefore, users can incrementally search for interested information without specifying all the search criteria. Detailed description of the hierarchical naming used in iShare can be found in [6]. The current implementation of the P2P network is built on a structured overlay network, Pastry [4]. Each node in the Pastry has a unique nodeId. When presented with a message with a key, Pastry nodes efficiently route the message to the node whose nodeId is numerically closest to the key,

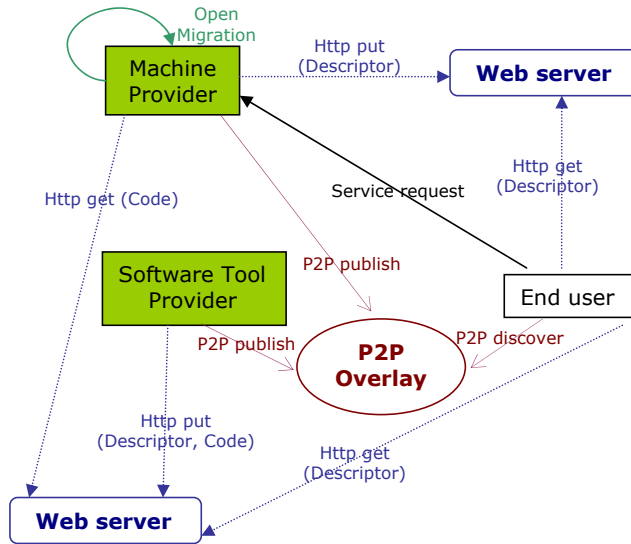


Fig. 4. HTTP and P2P communications in a roaming service request.

among all currently live nodes. A shared data item is distilled into standardized metadata and inserted with this $(route(msg, key))$ API. Requests for data are routed without requiring any knowledge of where the corresponding data items are stored.

A message is routed to the destination node whose Id is numerically closest to the hashing value of the message's key. To achieve fault tolerance, each publication operation creates a few replicas. The discussion of the replication and fault tolerance is beyond the scope of this paper, and details could be found in [6]. A local resource cache is designed to keep recently-used metadata. Successful end-user discovery operations will update the cache. Mechanisms to maintain cache consistency are described in [6]. A cached item is invalidated in three cases: 1) it expires after a live period; 2) any operation using the cached item fails; and 3) the user explicitly flushes the cache with a new discovery. The impact of caching on resource discovery latency is evaluated in Section 4.

3.4 Service Management: Management for Remote Resource Use

The concepts of remote resource use are realized in the service management module. Figure 5 presents the work flow graph for service management. Service management comprises three components: 1) resource allocation (advanced resource matching for roaming services); 2) resource usage monitoring and 3) automatic service portal creation. In this section, we will describe the initial designs for resource allocation and service portal creation. The second component is for future work.

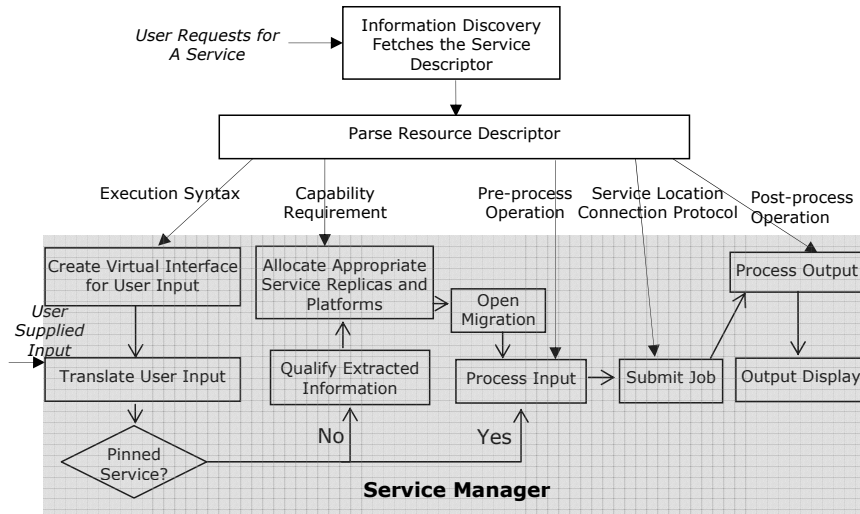


Fig. 5. Work flow graph for service management.

Resource Allocation for Roaming Services Roaming services are programs that pick the best combination of service replica and matching platform for every invocation. The resource allocation needs analyze (i.e., predict) the resource requirements of a given request and then match the requirements with appropriate resources. The resource requirements are a function of the user-supplied input, historical job execution profiles and capabilities of the execution platform. A learning algorithm described in [10] is used to characterize this relationship at the application level. The learning algorithm is expected to converge fast by taking advantage of the information gathered from peers' shared experiences. Decision for the matching also involves replication and caching management for service binaries, which is being developed in ongoing work.

Automatic Creation of Service Portals Service portals cannot be directly hard coded in iShare for two reasons: 1) the software implementation is usually tied to the local computing environment, and 2) the portal should be designed under the assumption that a run-time environment is self-contained. These issues can be addressed by creating service portals on-the-fly with resource descriptions of arbitrary software services and execution platforms. The service portal is mapped to the native job running environment at run-time. Service portals support plug-in access protocols by automatically generating job submission files and keeping track of site- and application-specific information.

4 Evaluation

4.1 Qualitative Evaluation

This section provides qualitative evaluations of the user model, and resource semantics components in iShare.

Accessibility The immediate advantage of accessing software services in iShare is the large savings in user efforts and computer resources. Typically, in order to make use of a tool without such an environment, an end user would first locate (e.g. using a web search engine) a downloadable tool that appears to fit the need [8]. To install the tool the user will need to obtain the appropriate machine resources and may need to perform software adaptations and configurations. In our experiences, this process can take from a few hours to several weeks. The unpredictability of this time is a critical disadvantage. In contrast, iShare benefits end users by providing access to already working software services and related documentation regardless of any client machine types. A typical service access time for first time users is nearly a minute, including authentication and navigating to the right tool.

Meanwhile, iShare benefits resource developers by providing a "usable" and efficient way to advertise new resources and solicit feedback. Adoption of the semantic description language allows providers to specify service semantics in a standardized interoperable manner [1] in addition to the syntax of service interfaces. The set of tools for creating and compiling resource descriptors make this process as easy as creating a HTML document. In our experience, first time users can learn and create a typical resource descriptor in a few minutes.

Scalability and Reliability Scalability issues are addressed by avoiding any central authorities in the system. Resource providers could choose any web site for posting resource descriptors and define site-specific rules to access resources. End users apply for access to physical resources by negotiating directly with the providers. The P2P network provides a media for communications between different communities. The scalability of the iShare system was tested by using resource discovery as an example, and the results are shown in the next section.

The P2P overlay is self-organizing and fault-tolerant itself, in the sense that a node join or leave does not fault the underlying DHT. Since the P2P network also works as an information repository in iShare, each published data is independently replicated on a set of neighboring nodes in the P2P overlay. Because of the randomness in the overlay construction, with high probability, the set of neighboring nodes over which the data are replicated are diverse in terms of geographic location and ownership.

4.2 Qualitative Evaluation

Of the four areas of contributions, as described in Section 3, this section provides quantitative evaluations of the information dissemination (in terms of the latency

of resource discovery) and the remote resource use techniques (in terms of the efficiency of job execution).

Efficiency of Resource Discovery We simulated the P2P based resource discovery, *iDiscover*, on a GT-ITM router network using the transit-stub model [3]. The size of the IP network is 1050 routers, 50 of which are used in transit domains and the remaining 1000 in stub domains. To test the scalability of *iDiscover*, we simulated several iShare testbeds with the number of nodes ranging from 500 to 10,000. We assume the iShare nodes are uniformly attached to the routers.

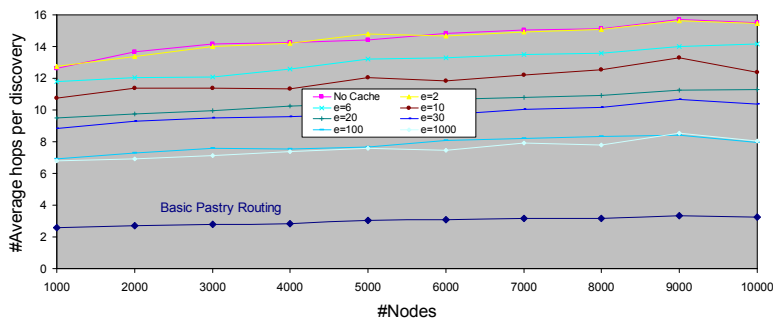


Fig. 6. Average number of hops per discovery. e is the normalized cache expiration time (cache expiration time/average request period).

In the experiment, we measured the effect of caching on discovery latency with different normalized cache expiration time e ($e = \text{cache expiration time} / \text{average request period}$). Each discovery operation starts from searching the root “Cyberlab”. One out of five nodes in the P2P network was randomly chosen to initiate the search request. Figure 6 plots the discovery response time with e ranging from 2 to 1,000. The figure shows that the discovery latency increases very slowly with the total number of nodes. We also see that with fair expiration time ($e = 6$), the response time is reduced by 10.37% on average compared to a discovery without local cache. The measurement results for cache hit rates are analyzed in [6]. Compared with the basic Pastry message routing, the resource discovery takes only a factor of 2-4 times longer, while supporting searches of resources with specific functionalities.

Efficiency of Remote Job Execution In this section, we compare the efficiency of remote job execution in iShare with that in SSH-based remote access. The goal is to show that iShare provides advanced network computing environment with acceptable costs in performance.

Remote access to computing services is commonly provided via explicit login to remote platforms (via REXEC, RSH and SSH). These mechanisms have several limitations [9], such as users having to manually identify and select remote

machines as well as being exposed to site- and platform-specific idiosyncrasies. iShare addresses these limitations by decoupling the computing environment perceived by users from the underlying physical environment. After getting the required access (e.g., account/password, certificate or public/private keys), end users can start an arbitrary job by interacting with the service portal. iShare translates and maps the user inputs automatically to the native service interface.

In iShare, the job execution includes the steps to fetch resource descriptor (from Web or local cache), parse the descriptor, create the service portal, authenticate, transfer input files, execute the program (including input processing and environment configuration) and process output. We chose a set of pinned services on a machine accessed via SSH. After getting accesses to these services, we ran them on iShare’s built-in SSH client and monitored the time spent at each of the above steps. To test the efficiency of the standard SSH, we manually coded Shell scripts to run each of these programs on the same remote machine with standard ssh and scp commands. A ssh agent is started manually to provide the authentication similar to iShare’s single sign-on mechanism (end users only need to input a password once). The efficiency difference between the public key authentication and password authentication is ignored in this experiment.

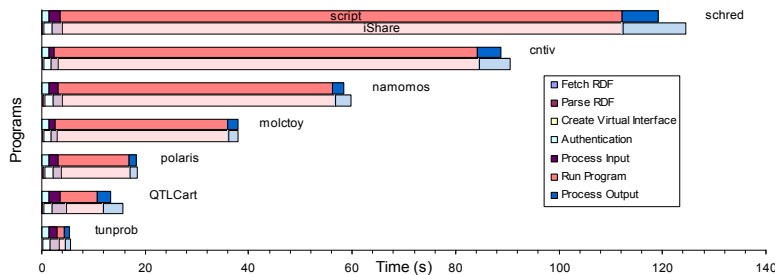


Fig. 7. Remote job execution through iShare and standard SSH.

Figure 7 shows the results for running a set of programs in iShare and with the manually-coded scripts. The time for parsing RDF descriptors and creating service portals is less than one second. The SSH-based job submission in iShare performs similar to the standard SSH protocol. The standard SCP protocol outperforms the iShare’s built-in file transfer protocols. The reason is that file I/O operations involved in SCP cause more overhead in iShare, because it is implemented with high level programming language (Java). Optimization of data transfer will be considered in future work.

5 Related Work

The research community is exploring a variety of approaches for constructing software infrastructures for Internet-sharing. On-going research can be divided into five categories with different foci and goals. The first category includes

global standardization efforts for grid computing, represented by GGF [23] and NMI [22]. Their objectives are to promote and support the development of Grid technologies via the documentation of technical specifications and the release of software packages. The focus of the second category is to provide application programmers a set of tools to harness “Grid” resources, e.g., to distribute massively parallel applications with message-passing. Examples of such work include Globus [11] and GridLab [14]. Work in the third category aims to develop “Grid-enabled” domain-specific applications. Active projects include EuroGrid [15], CrossGrid [17]. Work in the fourth category is geared towards providing end users (and resource providers) with the means to disseminate, access and use networked resources. Related work in this category includes active software web portals such as PUNCH [9], InVIGO [16] and NCSA-portals [18]; and web service techniques such as IBM WebSphere Application Servers [19]. The work in the fifth category is motivated by using P2P techniques to manage and share globally distributed resources. Active research includes large-scale file sharing systems [20] and compute cycle sharing [21].

The work described in this paper belongs to the fourth category, which differs from the other four categories by its user-orientation, its network accessibility to executable programs, its focus on “single-platform” rather than parallel applications and its support for generic instead of domain-specific programs. Within this category, we introduce the related work in end-user-oriented Internet-sharing systems and compare iShare in terms of in the design concept described in Section 2.

User Model: Software web portals provide end users direct access to unmodified software tools via standard Web browser. However, they don’t provide any open functionalities to resource providers. Most often, adding a new tool on a web portal has to involve the portal developers’ administration. iShare solves this problem by decentralized web-posting and P2P message routing. The ease of this publication process is comparable to creating a web page. Web service techniques enable users to build Web-based applications using preferred object model, programming language and platform. Service developers publish service descriptions to a central information location. This differs from iShare’s support for unmodified applications with no programming requirement and decentralized resource publication. iShare also differs from both web portals and web services in that it also integrates user functionality in common working environment.

Resource Semantics: Both web portals and web services target software application resources that are bound to fixed machine resources. iShare’s roaming services combine program and machine resources provided by different communities, thus offering a more flexible and open environment for resource sharing. Resource descriptions in web portals exist as static web pages containing documentations for specific programs. These web pages are manually maintained by portal administrators. Web services use WSDL for service interface specification and the specification is registered to a central UDDI registry. The RDF language adopted in iShare focused on semantics specification rather than syntax specification in WSDL. iShare’s resource description supports autonomy on the

definition of access protocols, while in web portals and web services, standard access protocols are required.

Information Dissemination: Resource discovery is not a critical issue in web portals, because all application resources are listed explicitly on web pages linked through the portal's main web page. In web services, users locate the services from the UDDI registry via SOAP. In contrast to the centralized structures in the two systems, iShare disseminates information via posting on individual web pages and registering to a P2P network. There is no central registry server or storage space in the whole iShare system. In addition to resource publication and discovery, iShare's dissemination mechanism is also used as a basis for exchanging user experiences and job execution statistics.

Remote Resource Use: The remote job execution in web portals and web services is supported by standardized protocols. They can be secured using HTTP basic authentication, HTTPS and SSL encryption, and digital signature. iShare supports plug-in protocols defined by individual providers via learning from the corresponding resource descriptions. The initial design of iShare supports commonly used authentication protocols such as SSH. Future work will extend iShare's security implementation to also support the Grid Security Infrastructure (GSI).

6 Conclusions

We have presented the design concepts and a implementation prototype of iShare - an Internet-sharing system built on P2P technology and the Web. iShare differs from related work in its user functionality for both providers and end users, its autonomy in defining and controlling local resources, its P2P-based information dissemination mechanisms and its support for roaming services and dynamic service portal creation. The evaluation results on resource discovery and remote execution confirm that iShare is able to deliver scalable and efficient Internet-based computing.

7 Acknowledgement

This material is based upon work supported in part by the U. S. National Science Foundation under Grants No. 9974976-EIA, 0103582-EIA, and 0429535-CCF.

References

1. K. Selcuk Candan, Huan Liu, Reshma Suvarna: Resource Description Framework: Metadata and Its Applications. ACM SIGKDD Exploration Newsletter, **3** (2001) 6-19
2. Resource Description Framework (RDF): Concepts and Abstract Syntax. <http://www.w3.org/TR/rdf-concepts/>

3. Ellen Zegura, Kenneth Calvert, Samrat Bhattacharjee: How to Model an Internet-work. Proc. IEEE INFOCOM, (1996)
4. A. Rowstron, P. Druschel: Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems. Proc. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), (2001) 329–350
5. B. Yang, H. Garcia-Molina: Designing a Super-peer Network. ICDE, 2003
6. Xiaojuan Ren, Zhelong Pan, Rudolf Eigenmann, Y. Charlie Hu: Decentralized and Hierarchical Discovery of Software Applications in the iShare Internet Sharing System. Proc. PDCS, 2004
7. Xiaojuan Ren, Rudolf Eigenmann: iShare – Internet Sharing of Programs, Machine and Data. Technical Report ECE-HPCLab-04203, High-Performance Computing Laboratory, Department of ECE, Purdue University, (2004)
8. Insung Park, Nirav H. Kapadia, Renato J. Figueiredo, Rudolf Eigenmann, José A. B. Fortes: Towards an Integrated, Web-executable Parallel Programming Tool Environment. Proc. Supercomputing Conference, 2000
9. Nirav H. Kapadia, Jose A. B. Fortes: PUNCH: An Architecture for Web-enabled Wide-area Network-computing. Cluster Computing, **2** (1999) 153–164
10. Nirav H. Hapadia, José A. B. Fortes, Carla E. Brodley: Predictive Application-Performance Modeling in a Computational Grid Environment. HPDC, 1999
11. I. Foster, C. Kesselman: Globus: A Metacomputing Infrastructure Toolkit. International Journal of Supercomputer Applications, 11(2), 1997
12. Litzkow M. J, Livny M, Mutka M. W: Condor - A Hunter of Idle Workstations Proc. ICDCS 1988, (1988) 104–111
13. R. Henderson, D. Tweten: Portable Batch System: External Reference Specification. Technical Report, NASA Ames Research Center, 1996
14. Gabrielle Allen, Kelly Davis, K. N. Dolkas, N. D. Doulamis, et. al.: Enabling Applications on the Grid - A GridLab Overview. International Journal of High Performance Computing Applications, 2003
15. Christian Hoppe, Pallas GmbH D. Mallmann, F. Julich: EUROGRID - European Testbed for GRID Applications. GRIDSTART Technical Bulletin, 2002
16. Adabala Sumalatha, Chadha Vineet, Chawla Puneet, Figueiredo Renato, et. al.: From Virtualized Resources to Virtual Computing Grids: The InVIGO System. Future Generation Computer Systems, 2004
17. Marian Bubak, Maciej Malawski and Katarzyna Zajac: The CrossGrid Architecture: Applications, Tools, and Grid Services. AxGrids, 2003
18. <http://www.ncsa.uiuc.edu/AboutUs/FocusAreas/ScientificPortalsExpedition.html>
19. IBM WebSphere Application Server. <http://www-306.ibm.com/software/webservers/appserv/was/>
20. KazaA. <http://www.kazaa.com/us/index.htm>
21. D. Anderson, et. al.: Internet Computing for SETI. ASP Conference Series, 2000.
22. NFS MiddleWare Initiative. <http://www.nsf-middleware.org/>
23. Global Grid Forum. <http://www.ggf.org/>