Bridging the Data Management Gap Between Service and Desktop Grids

Ian Kelley and Ian Taylor

Abstract Volunteer computing platforms have become a popular means of providing vast amounts of processing power to scientific applications through the use of personal home computers. To date, with little exception, these systems have focused solely on exploiting idle CPU cycles and have yet to take full advantage of other available resources such as powerful video card processors, hard disk storage capacities, and high-speed network connections. As part of the EDGeS project, we are working to expand this narrow scope to also utilize available network and storage capabilities. In this paper we outline the justifications for this approach and introduce how decentralized P2P networks are being built in the project to distribute scientific data currently on the Grid.

1 Introduction

For a number of years, volunteer computing environments have been extremely successful in leveraging the idle resources of personal computers. This has primarily been orchestrated through a donation system whereby private individuals allow their otherwise idle computers to be used by a third-party application for processing data. To date, major volunteer computing systems, such as the Berkeley Open Infrastructure for Network Computing (BOINC) [1][2], have focused solely on harnessing available CPU processing power, and have yet to take full advantage of the other available resource capabilities. With the sharp increases in consumer networking speeds and storage capacities over the past few years, utilizing idle network bandwidth to distribute data has become both a possible and attractive opportunity for volunteer computing.

Ian Kelley and Ian Taylor
School of Computer Science, Cardiff University, Cardiff, United Kingdom
e-mail: {I.R.Kelley,Ian.J.Taylor}@cs.cardiff.ac.uk

Enabling Desktop Grids for e-Science (EDGeS) [3][4] is an EU FP7 project that is setting up infrastructure and building software to enable the integration of Service Grids, or traditional Grid environments [5] generally composed of clusters and supercomputers, and Desktop Grid [1][6] systems, such as the popular volunteer computing project BOINC. When moving jobs between these two environments, and specifically when transferring a job from a Service Grid to a Desktop Grid, there is a need for some form of data management scheme to serve the files to participating Desktop Grid worker nodes.

One way to offload the central network needs that are created in this process and limit exposure to foreign hosts is to use a brokered data distribution mechanism. These brokers would act as a buffer between the two systems, receiving data from local hosts and managing the wider distribution challenges. Using peer-to-peer (P2P) techniques to implement such a system could be seen as a viable alternative to centralized data distribution. Not only would this reduce the Service Grid resources needed to integrate with a Desktop Grid, it could also mitigate the potential risk involved in transferring jobs to the Desktop Grid. By providing an intermediary layer, one is able to limit the number of peers to which a Service Grid node must distribute data. This can be further refined by applying project-based security criteria to govern the membership composition of the data brokers. For the Desktop Grid network, a P2P data distribution system would also allow current projects to take full advantage of client-side network and storage capabilities, enabling the exploration of new types of data-intensive application scenarios, ones that are currently overly prohibitive given their large data transfer needs.

There are many ways the aforementioned functionality could be implemented, ranging from BitTorrent-style networks [7][8], where there is a central tracker and all participants share relatively equal loads, to more customizable networks that allow for clients and service providers to be grouped. Custom-built solutions would have the advantage of facilitating different network topologies and data distribution algorithms. This allows for tailoring the network for the needs of an individual application, albeit with the disadvantage of increased development effort and code maintenance. In the case of BOINC, each of these approaches has its own distinct advantages and disadvantages, as explored in [9], especially when one takes into consideration the target user-community and their needs. Through the course of the paper we will show the data distribution work being undertaken in the EDGeS project as it advances towards interoperability between Service Grids and Desktop Grids. In particular, we will explore some of the requirements and varying scenarios that can appear in typical BOINC projects, outline the relative benefits of applying these new techniques, and give an overview of the data distribution software we are building for EDGeS.

The paper is organized as follows: section 2 gives background on the tools and related technologies involved; section 3 discusses EDGeS' data needs; section 4 gives an introduction of BOINC-specific requirements; section 5 overviews select design issues when applying P2P technologies to volunteer computing; section 6 proposes our decentralized data center approach; and section 7 concludes.

2 Background and related work

BOINC is currently the most widespread and successful volunteer computing Desktop Grid application ever, with over 50 distinct projects[1] and almost three million total computers from over 200 countries registered to date[2]. For data distribution, BOINC projects generally use a single centralized server or a set of mirrors. This centralized architecture, although very effective, incurs additional costs and can be a potential bottleneck when tasks share input files or the central server has limited bandwidth. Increasing the number of mirrors to accommodate increased loads puts extra administrative burden on the project organiser and can prove very time consuming to manage.

Popular and proven P2P technologies [10][11][12] such as BitTorrent, or commercial solutions like Amazon's S3[3] or Google's GFS [13], could be fairly effectively applied to provide for the data needs of BOINC, at least as they relate strictly to distribution. However, in the case of commercial products, there is a direct monetary cost involved, and for P2P systems like BitTorrent, the facility to secure or limit who is able to receive, cache, or propagate different pieces of information is generally limited or nonexistent. For example, BitTorrent, like many other P2P systems, has focused on ensuring conventional file-sharing features, such as efficient transfers, equal sharing and file integrity.

Desktop Grid environments have different requirements to general file-sharing P2P communities because security can become more of a complex issue than solely guaranteeing data validity (see section 5.1). In Desktop Grids, it can be a requisite that only certain amounts of data are shared with an individual peer. Communities can also be reluctant to introduce a system that would have peers directly sharing with one another, as it might have the potential (or perceived potential) to have security implications for clients as ports are opened for outside connectivity. It is therefore important not only for data integrity and reliability to be ensured, but also to have available safeguards that can limit peer nodes' exposure to malicious attacks. It is these types of requirements that has prompted our work to create a custom P2P network for data distribution that provides both client and server safeguards and stricter controls for project administrators as to what network participants receive and distribute data.

3 EDGeS' data needs

In the EDGeS project a job can be transferred from a Service Grid to a Desktop Grid. When this occurs, there is a need for some mechanism that either moves the

[1] There are over 50 known BOINC projects. At the time of this writing, the BOINC website has a list of 25 in which they have been in direct contact with: http://boinc.berkeley.edu/projects.php

[2] http://boincstats.com

[3] http://aws.amazon.com/s3

job's input files directly to the Desktop Grid workers, or exposes them on the Service Grid host for Desktop Grid peers to directly access and download.

At first glance, perhaps the easiest solution to enable access to the needed files would be to simply expose the data directly from the Service Grid file system. Such an approach would closely mimic the current functionality found in most Desktop Grid projects, where data is distributed to all participants from a central machine or through a set of known and static mirrors. This solution, although seemingly attractive in its simplicity, has many limiting drawbacks. For example, the Service Grid machine where the data is hosted might not be able to effectively serve the numerous peers making requests due to bandwidth limitations. Where previously the data was likely to be stored locally, allowing many processors access to it on a shared file system, now each peer which wishes to perform work must download an individual copy of the data to be analyzed. This can very quickly lead to a large drain on network bandwidth, especially in the case of larger files that need to be distributed to multiple workers.

In addition to raw resource usage concerns, there might also be security infrastructure and policies that would prevent access to local files from foreign and untrusted hosts. Anonymous access is generally not an issue for most BOINC projects, as they are able to have dedicated and network isolated data servers. This could, however, quickly become problematic, both technically and politically, if one tried to somehow bootstrap a BOINC data server onto a cluster or supercomputer to enable access to users' files. The situation is further complicated by the often complex software dependencies in existing Grid systems that make deploying yet another Grid service either not possible or at the very least unwelcome.

EDGeS requires a system that can adapt to varying input file sizes and replication factors without unduly stressing or exposing the Service Grid layer. This requirement will become increasing relevant as the EDGeS project moves beyond its test servers, which we can manage and configure, and begins connecting a wide range of EGEE [14] resources to different Desktop Grid systems. In this scenario, each of these federated Service Grid nodes will have different security infrastructures, internal policies, and network connectivity traits that would essentially render useless any system that required them to install additional software or adapt security policy. By pushing data to a P2P environment and offloading data distribution, Service Grid nodes could transfer the data distribution responsibilities, making the integration of Service and Desktop Grids more accessible.

4 BOINC requirements

As briefly mentioned in section 2, there are several dozen BOINC projects in operation. Every one of these projects shares a common thread with one another; each has a highly parallel problem able to be split into thousands of independent tasks that can be asynchronously processed. It is these properties that allow BOINC projects to exploit a Desktop Grid environment and utilize the numerous volunteer computing resources that are made available in the process. What isn't apparent, however, is that each of these projects can have vastly different levels of data intensity. This can manifest itself in the form of varying data input and output file sizes, changing replication facts, and different throughput requirements.

Given the dissimilar requirements for BOINC projects, there are many considerations one must take when thinking of applying a P2P system to the BOINC middleware. Even if a list of all the possible data distribution-related aspects were complied, various communities and application groups would have different priority rankings as to which are the most important for their individual circumstances (e.g., security vs. usability). The list we present here does not delve into the details of different project's data requirements, rather, it represents a few of the cross-cutting issues that are generally present in any BOINC-based project. Additional example areas to explore, which are not covered here, include topics such as data access speed, encryption, support for large data sets, and fuzzy query matching when searching for data. However, for the interests of simplicity and because the previously mentioned areas are currently being investigated, we have limited the scope explored here to the four issues listed below, which we believe to be key areas one must address when considering a P2P system that will be useful to the BOINC community as a whole.

Firewall and Router Configuration — Depending on an individual project's configuration, firewall and router issues could be problematic, with a general tradeoff between "punching holes" in clients' firewalls to be able to exploit their bandwidth and the security concerns and extra software development or configuration this demands. In volunteer computing projects it is especially important to provide a high level of security to participants. If NATs are bypassed, they need to be done in a secure and transparent (to the end-user) manner.

Malicious Users — The issue of which nodes are able to propagate data on the network, and therefore which ones will have the ability to inflict damage, will largely depend upon the individual policies of each hosting project. In the most restrictive case, only trusted and verified participants would be certified to propagate data. In looser security configurations, which allow for the exploitation of a larger pool of resources, security would have to be more flexible. Regardless of the decision, data signing can help to prevent any analysis of corrupted data. This makes network flooding the major concern, however, this can be limited relatively easily by implementing a ranking system to report misbehaving data providers.

Exploiting Network Topology — The ability to exploit network topology such as LANs and WAN proximity is a useful way to further limit the amount of necessary bandwidth to serve project files. The trade-off is generally that the looser the system becomes in its ability to adopt and utilize network proximity (such as providing caching nodes on LANs) the more exposed the network is to abuse and potential misconfiguration.

Integration with BOINC — It is important for any software that wishes to provide an added value to the larger BOINC community to have little or no impact on current operating procedures. Requiring external libraries or other similar dependencies could prove to be problematic and limit widespread uptake. The BOINC client is currently written in C++ and any successful add-on would most likely have to adapt to this requirement.

A more in-depth discussion of the above concerns and how they relate to the data distribution software being designed in EDGeS, along with a comparison to BitTorrent is discussed in [9].

5 Design considerations

Beyond the issues above, there are a number of general factors that become important when designing and deploying a data serving network across large-scale volunteer networks such as those in the BOINC community. For example, the size of the network can vary dramatically between the extremely popular BOINC projects and their less successful counterparts. Aside from sheer network size, different projects will have varying data input and output file sizes, with some projects having a peer transfer over a gigabyte per month while others require only a fraction of this amount. For each project within BOINC, these factors are slightly different, and the optimal network setup for one project might not be very efficient for another. These differences make designing an optimal network for BOINC as a whole a challenging task. Yet as shown in [15], the application of a P2P network layer would allow many additional and unused network and storage resources to be leveraged by BOINC projects without sacrificing necessary processing power.

In section 6, we introduce the data distribution software we are in the beginning phases of developing. However, before talking of the implementation and design specifics, it is useful to further expand upon the requirements listed in section 4 and discuss some of the cross-cutting security issues shared between BOINC projects. This will help to set the stage for the proposed architecture.

5.1 Security aspects

When building a P2P network for volunteer computing, there are a number of security requirements beyond the traditional notion of simply ensuring file integrity. Due to the volatile and insecure nature of BOINC networks, a product of their open participation policies, there can be reason to enforce limitations on which entities are allowed to distribute and cache data. When opening the data distribution channels to public participation, security can become a concern. In this context, the area of security can be roughly split into the following two distinct realms: *user security* and *data security*.

User security refers to the ability to protect the "volunteers" in the network from any harm that could possibility be inflicted on their machine by another network participant. This is a very important issue when one is within the realm of volunteer computing, as the resources are typically donated home computers that likely contain volunteers' important personal information and documents. In the case of a security breech in which these volunteer resources were compromised by some malicious entity, the potentially fallout could be enormous. Fear of a harsh backlash has been one of the limiting factors to the incorporation of standard P2P technologies into the BOINC middleware. Even in the event where no actual security breech takes place, *requiring* peers to share data with one another via P2P protocols, such as BitTorrent, which enforces sharing, could have the down-side of alienating potential volunteers. This could result from any number of factors, ranging from a volunteer's unwillingness to donate network resources (perhaps due to bandwidth requirements from other computers on the same network or a metered data connection) to misconceived public perception that associates peer-to-peer technological implementations with some of the more controversial uses of the technology.

It is necessary to be cognizant of the fact that the BOINC community relies upon volunteers to function, and any "peer-to-peer" data distribution scheme that is implemented must allow for users to opt-out if they do not wish to share their bandwidth or storage capacity. Even in the instance where users have opted to share data, a generally high level of consideration has to be given to ensure that their computers are adequately protected from attacks. In current BOINC environments this is solved by having a centralized, and presumably non-hostile, authority that distributes both executables and data. Although even in this scenario, there are still chances that the servers could be compromised, or that the executables distributed have inherent security flaws, this is generally a very minimal risk and would be a consequence of actions of the application stakeholders, not third-party unknown distributers. It is these considerations and requirements that make applying P2P protocols such as BitTorrent, which enforce tit-for-tat sharing, problematic.

Data security can be a complex matter. First, there is the issue of file integrity, which we will not go into in detail here, mainly because there are many well-known and suitable techniques to validate the authenticity of files, such as hashing and signing. More interesting and novel is investigating security schemes for which to select and distribute data-sets to peers. When looking at this issue in more detail, it can further be broken down into two broad subject areas: *authentication* and *authorization*.

Authentication is the verification process by which an entity identifies itself to others and gives evidence to its validity. Public key infrastructure (PKI) [16] is a proven tool that can be fairly effectively applied for performing peer identity authentication. In the simplest case, this can be done by having a central authority (i.e., the BOINC manager) sign and issue either full or proxy certificates to those it deems trustworthy enough to distribute data on its behalf. When another peer on the network contacts this "trusted" entity, it can use the public key from the centralized BOINC manager to verify the authenticity of the trusted peer. This process can likewise be performed in reverse, provided clients are also issued certificates, as a means for the data distributers to validate the identity of the clients and verify they have the proper credentials to retrieve data. The process of using certificates for mutual authentication can be a fairly effective solution that would provide individual peers with certainty that the host they are retrieving data from has been delegated the proper authority and visa versa. More interesting use-cases that provide for interaction between multiple virtual organizations (VOs) and hierarchal delegation (e.g., certificate-chaining and cross-certification agreements) can be derived from this simple arrangement, but are beyond the scope of this paper [17][18].

Authorization is a much more interesting question than authorization. This is primarily because there are standardized techniques for authentication that can be widely applied to many different applications with little or no modification. Authorization conversely is application-specific, differing with each individual application's unique needs and authority structures. Although there are tools to help define authorization policies and enforce them [19], the policies themselves will be different with each application.

At the most basic level of authorizing select peers to cache and distribute data as they see fit, authorization is very simple and should not prove problematic. For example, it is possible to issue special certificates to the data cachers (as mentioned in the authentication section) that allows them to validate as data distributors. However, when more dynamic and customizable queries are needed, such as finer-grained control over what data can be propagated by individual data cachers, the issue of authorization becomes more complex. When this occurs, the issue requires more due diligence, and any scheme that goes beyond a simple yes/no query must be customized specifically to the target environment, in this case, not only BOINC, but potentially each target community.

6 Proposed architecture: *ADICS*

The peer-to-peer **A**rchitecture for **D**ata-**I**ntensive **C**ycle **S**haring (ADICS) [20] is an effort to build a P2P software layer that can be used by scientific applications, specifically those engaged in volunteer computing, to distribute, manage, and maintain their data. Therefore, the core infrastructure of ADICS is being built with the needs of a scientific user and application in mind. As a software development package being supported by the EDGeS project, ADICS is taking very close consideration of the issues raised in section 3. It should be noted, however, that the core infrastructure is being architected to be fairly application agnostic and should therefore be applicable to other applications that want to distribute data in a peer-to-peer environment.

Based upon the previously mentioned security and user requirements, the network we have chosen to implement at this time is an architecture that has three distinct entities: *workers*, *data centers*, and *data providers*. In this 3-tiered, bridged architecture, the *data provider* pushes files to the *data center* overlay network, which self-organizes using P2P techniques to propagate data amongst itself. This data center layer then serves pull requests for data from the *workers*. Figure 1 gives a generalized overview of how the different components in ADICS relate to one another after the initial discovery phase. During discovery, a worker node would send a request to known access points in the data center overlay network and retrieve an updated list of connection points from which it can harvest data. If this process fails, likely due to a stale list of hosts, the worker node is able to contact the static data provider to request a new data center reference. Subsequent requests for data are made to the data center layer, and the worker is then able to contact one or more centers for downloading.

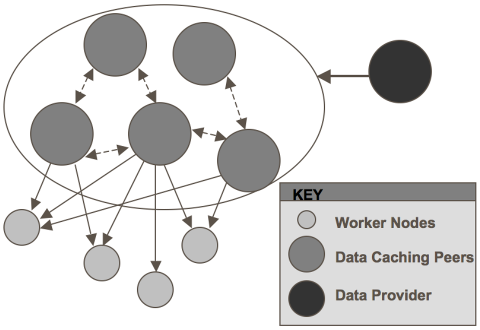

Fig. 1 ADICS schematic of the organization of the distributed entities that shows a data provider serving data to the caching layer, which is, in turn, distributed to the worker nodes.

The reasoning behind this is to allow for only a subset of peers that match certain performance and security thresholds to share data with the rest of the network. By implementing an opt-in and (for the moment centrally) validated system for data sharing, many of the security considerations (see sections 4 and 5) such as router configuration, automatically opening ports, and rouge hosts providing data can be marginalized. In this scenario, the *data center* subset of peers on the network act as "true peers" in the sense that that both send and receive on an equal standing with their data center neighbors, however, they act solely as servers to the *workers*. One benefit of this approach is that, *workers* continue to operate relatively unchanged from their previous BOINC working conditions, with the relatively minor addition of a distributed data lookup.

Secure data centers are the name we have given members of the super-peer overlay that are engaged in data sharing. The *secure* aspects becomes apparent when constraints are later put upon the registration phase, thereby restricting the set of peers that are allowed to propagate data. Policy decisions as to which participants, if any, are allowed to host and redistribute data would be made by each individual BOINC project, with ADICS providing the base infrastructure to aide the process as well as a default implementation. Once the general tools are in place, more complex scenarios can be explored that go beyond simply restricting data center membership. For example, constraints could be introduced to govern the relative sensitivity of data and retention policies. Adding these new types of functionality would allow for more advanced use-cases, albeit with the additional costs of software and network complexity.

Based upon the preliminary results of [21] and the arguments presented here, it is our belief that decentralized data centers can prove to be both valid and useful solutions to distributing data in Desktop Grid environments. There is, however, a tradeoff between functionality and complexity that needs to be adequately addressed and balanced if such technologies are to be adopted by production environments such as BOINC.

6.1 Prototype development

Initial development in ADICS was based upon the Peer-to-Peer Simplified (P2PS) middleware [22], a generic P2P infrastructure and JXTA [23]. Although both P2PS and JXTA provide generic tools for building super-peer networks, they proved to be limiting either in their ability to scale or to form role-based groups where the developer can explicitly form the topology and control message relaying without major modifications. Specifically, P2PS and JXTA have been abandoned because of two main reasons. First, neither allow the fine-grained access controls needed for the data layer. Second, there are no caching policies in either system for data rather than metadata (adverts or queries). Therefore, the data layer would essentially have to be built from scratch, meaning that the benefits of either system are reduced to providing their respective P2P abstractions. It was therefore decided that the benefits of using these systems were far outweighed by the drawbacks of the additional dependencies they placed upon the end-user, and their increased complexity.

The current focus is on implementing a specific system that fits into the current BOINC messaging layer, yet is generic enough to be applied in a number of different ways. To this end, we are developing a prototype that will help to define the entities and evolve the design of the network and its messages. This will allow us to validate the selected topology and show that it is useful to solve the security and data propagation issues introduced earlier in this paper.

Figure 2 gives a general overview of the different network interactions that a worker has in the current prototype. In order to enable the prototype to function independently of BOINC and speed development, we have implemented a very basic work generation entity (the Network Manager) that generates work units to fulfill client requests and begin the data retrieval cycle. It should be noted that in the prototype available at the time of this writing, the data center layer is fed by the central repository, and does not self-propigate data amongst itself.

- **(1)** A worker requests a *WorkUnit* from a known Network Manager server.
  (1b) The worker receives the response and extracts a list of *DataQueries*, which contain information on how to identify the job's data dependencies. Currently this is a unique ID, however, it could also be a more sophisticated query.
- **(2)** The worker contacts a Data Lookup Service, and provides it with one or more *DataQueries*. Currently this service is known and centralized, however, there are plans to decentralize it and provide it as a service on the Data Center layer.
  (2b) The Data Lookup Service attempts to match each *DataQuery* to a real file

mapping and, if successful, returns a *DataPointer* for each *DataQuery*. The *DataPointer* contains a list of Data Centers that are believed to have he file, as well as any associated metadata about the file that is available.

- **(3)** For each *DataPointer*, the worker extracts the location of one or more Data Centers that are believed to host the file. The worker then directly connects to one or more (currently one) Data Center for retrieval.
  (3b) The Data Center retrieves the file from its local disk space and sends it to the worker.

In its current implementation, the prototype is very similar in nature to how early Napster worked, with a central metadata server keeping track of where data is located. The Data Lookup Service is also reminiscent of a BitTorrent tracker, which performs essentially the same functions on behalf of a single file.

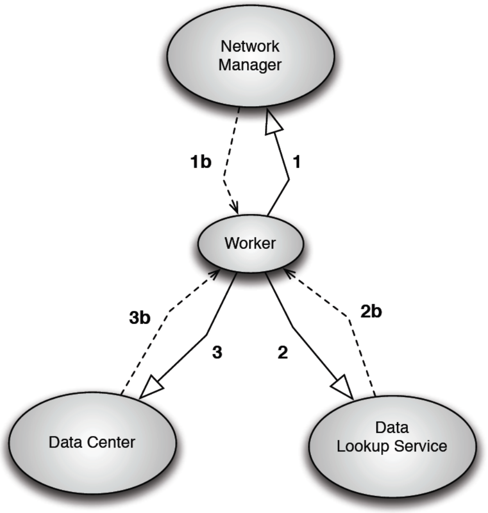

Fig. 2 Diagram showing the basic flow of messages from the worker to the various other network entities.

One of the main differences between these previous systems and the described here data center scheme is the potential for the addition of security criteria which restricts the data service layer to a subset of the available peers. The next step in ADICS development is to decentralize the data lookup facility through implement self-organizing traits on the data center layer, and to add security constraints on the propagation and lookup of data. At the time of this writing, this is currently a work-in-progress.

One of the clear issues raised in section 4 was that of integration with the existing BOINC software stack. ADICS is currently being designed and built in Java, which naturally creates a client dependency on a JRE. As mentioned in section 4, this can pose a problem when trying to later integrate ADICS into the BOINC client layer. Two possible solutions to this problem are: *(1)* add a JRE to the required software to run BOINC, which could severely limit adoption of ADICS; and, *(2)* create a C++ implementation of ADICS for *workers* that provides download capabilities. This would allow for the core discovery and data propagation layer to be left in Java. By doing so, we would limit the JRE requirement to nodes that wish to operate as data centers. Current design and plans for ADICS are pursuing option *1*, in order to build a working system as quickly as possible. The necessity of option *2* will be reassessed later based upon feedback from the BOINC user-community.

It should be noted that the software discussed here is currently under heavy development and is expected to evolve as new challenges are encountered. As part of the EDGeS project deliverables, the first public prototype of ADICS will be released in December 2008. This prototype will have functionality that allows for the propagation, caching, and sharing of data, using the decentralized data center architecture outlined here.

6.2 Research theories verified through simulation

Much work in the few months since EDGeS started has focused on using simulation tools, such as NS-2 [24] and [21]. The goal of this work is to verify the ideas presented here and define the general structure of a network that supports EDGeS' needs and attempts to model the transfer of typical sized data files and network loads for current BOINC projects, such as Einstein@HOME. This is currently being achieved through the construction of reusable network topologies that represent standard home Internet connections in systems like NS-2. We are applying these topologies to the ADICS prototype code though the third-party AgentJ libaries [25] that allows for the attachment of Java entities to NS-2. Currently we are using this to represent the behaviour of the central data repository (i.e., the BOINC server), the data caching layer, and the connected peers, while taking into consideration parameters such as the network links and the underlying protocols.

Previous simulation work [21] was also successfully undertaken to explore a more general cycle-sharing paradigm and the suitability of a data center approach for caching job input files in distributed environments such as BOINC. Although the work presented in that paper was more generalized, the fundamental "dynamic caching" and data distribution aspects are consistent with the ones presented here. These prior simulations were very useful in helping to shape the general discussion and move towards the more specific network simulations that are taking place now. We are continuing to refine those simulations to mimic real-life use-cases, in addition to the NS-2 models previously mentioned.

7 Conclusion

In this paper we have discussed some of the data issues that arise in the EDGeS project as Service Grid jobs are migrated to a Desktop Grid environment. One low-cost way to provide the needed bandwidth and data storage to support this scenario is to exploit client-side network capacities in a P2P-type system for distributed data sharing and propagation. The brokered approach outlined here, ADICS, can provide a happy medium between a "true" P2P system implementation that treats participants relatively equally and the current static mirroring being used by BOINC projects. ADICS has three main entities: data providers that push data onto the network; data centers (or cachers) that conform to some security requirement that allows them to propagate data on the data providers' behalf; and, data consumers (or workers) that execute jobs and request input files from the data center layer. The architecture outlined here is currently being pursued within the EDGeS project. It is hoped this paper will further the discussion on the applicability of P2P technologies in the scientific community and encourage others to explore it as a valid and useful approach for data distribution.

Acknowledgements This work was supported by the CoreGRID Network of Excellence, the Center for Computation & Technology at Louisiana State University, EPSRC grant EP/C006291/1, and the EU FP7 EDGeS grant RI 211727.

References

1. "BOINC - Berkeley Open Infrastructure for Network Computing." [Online]. Available: http://boinc.berkeley.edu/
2. D. P. Anderson, "BOINC: A System for Public-Resource Computing and Storage," in *5th IEEE/ACM International Workshop on Grid Computing*, Pittsburgh, USA, November 2004, pp. 365–372.
3. "Enabling Desktop Grids for e-Science." [Online]. Available: http://www.edges-grid.eu
4. Z. Balaton, Z. Farkas, G. Gombas, P. Kacsuk, R. Lovas, A. C. Marosi, A. Emmen, G. Terstyanszky, T. Kiss, I. Kelley, I. Taylor, O. Lodygensky, M. Cardenas-Montes, G. Fedak, and F. Araujo, "EDGeS: The Common Boundary Between Service and Desktop Grids," in *To be published in a special volume of the CoreGRID Springer series.*, 2008.
5. I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," *Int. Journal of Supercomputing Applications*, vol. 11, no. 2, pp. 115–128, 1997.
6. F. Cappello *et al.*, "Computing on Large-Scale Distributed Systems: XtremWeb Architecture, Programming Models, Security, Tests and Convergence with Grid," *Future Generation Computer Systems* , vol. 21, no. 3, pp. 417–437, 2005.
7. B. Cohen, "Incentives Build Robustness in BitTorrent," in *Workshop on Economics of Peer-to-Peer Systems (P2PEcon'03)*, Berkeley, CA, June 2003.
8. "Bittorrent." [Online]. Available: http://www.bittorrent.com/
9. F. Costa, I. Kelley, L. Silva, and I. Taylor, "Peer-To-Peer Techniques for Data Distribution in Desktop Grid Computing Platforms," in *To be published in a special volume of the CoreGRID Springer series.*, 2008.
10. J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "OceanStore: an architecture for global-scale persistent storage," *SIGPLAN Not.*, vol. 35, no. 11, pp. 190–201, 2000.
11. S. S. Vazhkudai, X. Ma, V. W. Freeh, J. W. Strickland, N. Tammineedi, and S. L. Scott, "FreeLoader: Scavenging Desktop Storage Resources for Scientific Data," in *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*. Washington, DC, USA: IEEE Computer Society, 2005, p. 56.
12. I. Clarke, S. G. Miller, O. Sandberg, B. Wiley, and T. W. Hong, "Protecting Free Expression Online with Freenet," *IEEE Internet Computing*, pp. 40–49, January, February 2002.
13. S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 29–43, 2003.
14. "EGEE: Enabling Grids for E-science in Europe." [Online]. Available: http://public.eu-egee.org

15. D. P. Anderson and G. Fedak, “The computational and storage potential of volunteer computing,” in *CCGRID ’06: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 73–80.
16. “IETF Public Key Infrastructure Working Group.” [Online]. Available: http://www.ietf.org/html.charters/pkix-charter.html
17. K. Berket, A. Essiari, and A. Muratas, “PKI-based security for peer-to-peer information sharing,” *Peer-to-Peer Computing, 2004. Proceedings. Proceedings. Fourth International Conference on*, pp. 45–52, 25-27 Aug. 2004.
18. J. E. Altman, “PKI Security for JXTA Overlay Networks,” IAM Consulting, Inc., Tech. Rep., 2003.
19. T. Barton, J. Basney, T. Freeman, T. Scavo, F. Siebenlist, V. Welch, R. Ananthakrishnan, B. Baker, M. Goode, and K. Keahey, “Identity Federation and Attribute-based Authorization through the Globus Toolkit, Shibboleth, GridShib, and MyProxy,” in *NIST PKI Workshop*, April 2006.
20. “Peer-to-Peer Architecture for Data-Intensive Cycle Sharing (ADICS).” [Online]. Available: http://www.p2p-adics.org
21. P. Cozza, I. Kelley, C. Mastroianni, D. Talia, and I. Taylor, “Cache-enabled super-peer overlays for multiple job submission on grids,” in *Grid Middleware and Services: Challenges and Solutions*, D. Talia, R. Yahyapour, and W. Ziegler, Eds. Springer, 2008, to appear.
22. I. Wang, “P2PS (Peer-to-Peer Simplified),” in *Proceedings of 13th Annual Mardi Gras Conference - Frontiers of Grid Applications and Technologies*. Louisiana State University, February 2005, pp. 54–59.
23. “Project JXTA.” [Online]. Available: http://www.jxta.org/
24. “The Ns2 Simulator.” [Online]. Available: http://www.isi.edu/nsnam/ns/
25. I. Taylor, I. Downard, B. Adamson, and J. Macker, “AgentJ: Enabling Java NS-2 Simulations for Large Scale Distributed Multimedia Applications,” in *Second International Conference on Distributed Frameworks for Multimedia DFMA 2006*, Penang, Malaysia, 14th to 17th May 2006.