UNIVERSITY OF CALIFORNIA, SAN DIEGO

Coordinated Resource Management for Guaranteed High Performance and Efficient Utilization in Lambda-Grids

A dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in

Computer Science

by

Nut Taesombut

Committee in charge:

Professor Andrew A. Chien, Chair
Professor Walter A. Burkhard
Professor Chung-Kuan Cheng
Professor Rene L. Cruz
Professor George Papen

2007

The dissertation of Nut Taesombut is approved, and it is acceptable in quality and form for publication on microfilm:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2007

DEDICATION

For their caring, support and encouragement,

I dedicate this dissertation to all my family and teachers.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

There are many people who I am deeply indebted to and so privileged to know, and without them this dissertation wouldn't have been a reality. I cannot name them all here, and cannot hope to adequately repay them.

First and foremost, I sincerely thank my advisor, Professor Andrew Chien, for his endless support, patience and overall guidance through various stages of my Ph.D. study. I admire him for always believing in me, teaching me so many aspects of life, and challenging me to improve myself. I considerably benefited from discussions with him, and his inputs on research ideas have greatly contributed to this dissertation. I would also like to thank my committee members, Professor Walter Burkhard, Professor Chuang-Kuan Cheng, Professor Rene Cruz and Professor George Papen, for their eagerness to spend time learning my research and give me invaluable comments.

I am grateful to the entire faculty, staffs, and students in the OptIPuter project for providing me the opportunity to conduct the cutting-edge research in advanced optical networking and distributed computing systems. I am indebted to countless number of people for their time and support for software, hardware and system administration to make the research presented here feasible.

I thank all my colleagues at UCSD for their support through these years. I am fortunate to have opportunities to work with extremely talented and high-spirited graduate students. I thank everyone in the CSAG group, especially Ryan Wu, Dionysios Logotheis, Richard Huang, Frank Uyeda, Huaxia Xia, Justin Burke, Eric Weigle, Ju Wang, Xin Liu, Luis Rivera, Adam Brust, Troy Chuang, Alex Olugbile, Jerry Chou and Ryo Sugihara. I thank them for their friendships, compassion and valuable discussions that helped improve my research. I especially thank Yang-Suk Kee for his generous support and guidance on my research and many aspects of life.

I thank all my friends in San Diego, especially Thawee Techathamnukool, Chanathip Namprempre, Kitirat Panupong, Nada Wasi, Yuenyong Songsiridej, Martha Stacklin and David Hutches, for making my life here so worthwhile and enjoyable. Without their continuing support and encouragement I wouldn't make it through all these years. I am also deeply indebted to my teachers and friends at the San Fran Dhammaram Temple and KPY Buddhist Monastery for helping me understand the ultimate purpose of life and encouraging me to do the right things.

Last but not least, I want to thank my parents and sister for their never-ending love, support and guidance. My father has been the greatest influence on my life. His determination and courage are rare and never cease to impress me. My mother's love and generosity to her children are unmatched and have always inspired to be the person that I am today.

Part of Chapter 1, Chapter 3 and Chapter 4 is published as “Distributed Virtual Computer (DVC): Simplifying the Development of High Performance Grid Applications” by Nut Taesombut and Andrew A. Chien in the proceedings of the Workshop on Grids and Advanced Networks (GAN’04), April 2004. The dissertation author was the primary researcher and co-author of this paper.

Chapter 6, in part, is published as “Evaluating Network Information Models on Resource Efficiency and Application Performance in Lambda-Grids” by Nut Taesombut and Andrew A. Chien in the proceedings of ACM/IEEE International Conference on High Performance Computing and Communication (SC’07), November 2007. The dissertation author was the primary researcher and co-author of this paper.

Chapter 7, in part, is published as “Collaborative Data Visualization for Earth Sciences with the OptIPuter” by Nut Taesombut, Xinran Wu, Andrew A. Chien, Atul Nayak, Bridget Smith, Debi Kilb, Thomas Im, Dane Samilo, Graham Kent and John Orcutt in Journal of Future Generation Computer Systems, Vol. 22(8), October 2006. The dissertation author was the primary researcher and co-author of this paper.

The work presented here was supported in part by the National Science Foundation under awards NSF Cooperative Agreement ANI-0225642 (OptIPuter), NSF CCR-0331645 (VGrADS), NSF ACI-0305390, and NSF Research Infrastructure Grant EIA-0303622. Support from the UCSD Center for Networked Systems, BigBangwidth, and Fujisu is also gratefully acknowledged.

VITA

2000	Bachelor of Engineering, Chulalongkorn University, Thailand
2002-2003	Teaching Assistant, Department of Computer Science and Engineering, University of California, San Diego
2003	Master of Science, University of California, San Diego
2003	Research Intern, AT&T Research Lab
2003-2007	Research Assistant, University of California, San Diego
2007	Doctor of Philosophy, University of California, San Diego

PUBLICATIONS

N. Taesombut and A. A. Chien. "Evaluating Network Information Models on Resource Efficiency and Application Performance in Lambda-Grids," in *Proceedings of ACM/IEEE International Conference on High Performance Computing and Communication (SC'07)*, November 2007.

N. Taesombut and A. A. Chien. "Evaluating the Impacts of Network Information Models on Applications and Network Service Providers," in *Proceedings of IEEE International Symposium on High Performance Distributed Computing (HPDC'07)*, June 2007

R. Singh, N. Schwarz, N. Taesombut, et al. "Real-time Multi-scale Brain Data Acquisition, Assembly, and Analysis Using End-to-End OptIPuter," *Journal of Future Generation Computer Systems*, Vol. 22(8), October 2006. pp. 1032-1039.

N. Taesombut, X. Wu, A. A. Chien, et al. "Collaborative Data Visualization for Earth Sciences with the OptIPuter," *Journal of Future Generation Computer Systems*, Vol. 22(8), October 2006. pp. 955-963.

N. Taesombut, F. Uyeda, A. A. Chien, et al. "The OptIPuter: High-Performance, QoS-Guaranteed Ntwork Service for Emerging E-Science Applications," *IEEE Communication Magazine*, Vol. 44(5), May 2006. pp. 38-45.

N. Taesombut and A. A. Chien. "Distributed Virtual Computer (DVC): Simplifying the Development of High Performance Grid Applications," in *Proceedings of the Workshop on Grids and Advanced Networks (GAN'04)*, April 2004.

A. Boldyreva and N. Taesombut. "Online Encryption Schemes: New Security Notions and Constructions," in *Proceedings of the RSA Conference 2004 Cryptographers' Track (CT-RSA'2004)*, February 2004.

N. Taesombut, R. Huang, and P. V. Rangan. "A Secure Multimedia System in Emerging Wireless Home Networks," in *Proceedings of the 7th IFIP Conference on Communication and Multimedia Security*, October 2003.

N. Taesombut, V. Kumar, R. Dubey, P. V. Rangan. "A Secure Registration Protocol for Media Appliances in Wireless Home Networks," in *Proceedings of the 2003 IEEE International Conference on Multimedia and Expo*, July 2003.

FIELDS OF STUDY

Major Field: Computer Science and Engineering

Studies in Distributed Computing and Advanced Optical Networking
Professor Andrew A. Chien, University of California, San Diego

ABSTRACT OF THE DISSERTATION

Coordinated Resource Management for Guaranteed High Performance and Efficient Utilization in Lambda-Grids

by

Nut Taesombut

Doctor of Philosophy in Computer Science

University of California, San Diego, 2007

Professor Andrew A. Chien, Chair

Emerging configurable optical networks and Grid computing create intriguing opportunities for new application capabilities and resource efficiencies. Applications can exploit dedicated, high-speed optical circuits to tightly interconnect remote resources on-demand, and achieve high quality of service. However, they must contend with the complexity of highly distributed and heterogeneous resource environments. In addition, network configurability presents unique challenges, adding the complexity of planning configurations to that of traditional end resource management.

To enable efficient and simple development of high performance applications, this dissertation proposes the Distributed Virtual Computer (DVC), a novel integrated architecture for

managing configurable networks and wide-area resource sharing. The DVC allows an application to describe and acquire a combined set of communication and end resources, and then automatically manages them for guaranteed, high performance. Such an integrated approach enables coordinated resource management improving both application capabilities and resource efficiencies.

In this framework, a key challenge is selecting appropriate sets of resources for individual applications. We formulate the selection problem, explore several approaches, and evaluate each via simulation. Best performance is achieved by techniques that combine the selection of communication and end resources. Such approaches produce high-quality solutions both for application performance and for network efficiency, and scale well for large resource environments. This enables an online service where applications can request and acquire high-quality resources quickly on-demand.

In a multi-domain network, a critical tension exists between service providers who are business competitors. As a result, controlled information sharing is required that balances their competitive positions and enables efficient resource selection. We characterize the network information that could be shared between providers and assess how individual information affects applications and service providers. Our results suggest providers should share their internal information as it can improve their resource efficiencies and application performance.

We implement a DVC system software prototype and present experimental results with real scientific applications and optical networks. We demonstrate our prototype enables the simple configuration of collaborative data visualization environments that can be flexibly run on different physical resource configurations. Additionally, the applications are able to exploit dedicated optical circuits on-demand and efficiently utilize the network capacity.

Chapter 1. Introduction

1.1 A Novel Opportunity of the Lambda-Grid

Large-scale e-science applications such as scientific data distribution and sharing [1], computational steering [2] and collaborative data visualization [3], are emerging in virtually every scientific field, including earth science [4], neuroscience [5], oceanography [6], nuclear physics [7] and astronomy [8]. They can be used to simulate and analyze highly complex systems, but require massive aggregations of computing, storage and visualization resources. Because such applications involve teraflop-scale computation, petabyte-scale distributed data collections and wide-area collaboration, their resource requiremenets cannot be satisfied within a single organization.

Grid computing [9] has emerged as a technology that enables coordinated resource use across geographical locations and organizations. Grid resources, such as distributed computing clusters, petabyte data stores and other high-end scientific instruments, can be securely shared through a Virtual Organization (VO) [10]. A VO is a set of relationships and sharing policies that grant users access to resources across traditional organizational boundaries. This enables scientific researchers to exploit far greater computing, storage and collaboration capabilities. Examples of today's important Grid projects include TeraGrid [11], iVDGL [12], GriPhyN [13] and EU-DataGrid [14].

Today, the phenomenal amount of data being produced, collected and processed by these Grids poses a significant challenge to e-science applications. For example, the Biomedical Informatics Research Network (BIRN) [5] is a National Institutes of Health (NIH)-funded project to support global collaboration of medical neuroscience research. The NIH estimates the current total data held by BIRN sites is 10 petabytes, and this figure is likely to increase 1,000x in the next decade. In addition, EarthScope is the National Science

Foundation (NSF)-supported project [4] studying the structure and evolution of the Earth's crusts in North America. EarthScope has deployed modern observational sensors over a span of Alaska and other parts of the U.S. that produce 40 terabytes of seismic data annually. Supporting real-time scientific data analysis and sharing such large quantities of information requires extremely high bandwidth and reliable network service. For instance, a typical scientific dataset, such as a high-resolution 3D image of scanned mouse brains, can be as large as 100's of gigabytes. Achieving real-time, remote data acquisition, visualization and exploration requires a data transfer rate of several gigabits per second as well as bounded communication jitter and delay. With emerging scientific applications' tremendous increase in demand for bandwidth and QoS traffic characteristics, the traditional best-effort Internet is insufficient.

Recent advances in optical transmission and distributed network control plane are producing novel wide-area networks with dramatically increased bandwidth and controllable performance properties. A key driver is the Dense Wavelength Division Multiplexing (DWDM) technology which enables large numbers of wavelengths (or lambdas) to be carried over a physical fiber thus improving available bandwidth by several folds. Specifically, each lambda can carry a signal at a bit rate of ten gigabits per second, and the aggregate throughput of a fiber (100's of lambda's) can be up to several terabits per second. Further, a maturing network control plane is enabling dynamic provisioning of these lambdas. A high-speed optical circuit (or lambda) can be configured on-demand to tightly interconnect remote resources in seconds and optimize application data flows. Because each lambda is independent and congestion-free, network properties such as bandwidth, jitter and delay can be planned and controlled. Dynamic network provisioning not only allows individual applications to obtain dedicated use of real "private" networks, but also enables efficient use of communication resources (e.g., switch ports and wavelengths). Examples of advanced lambda

network facilities include OptIPuter [15], National Lambda Rail [16], DRAGON [17], CHEETAH [18], Global Lambda Interchange Facility (GLIF) [19], CANARIE's CA*net 4 [20], and NetherLight [21].

A Lambda-Grid [22] is a Grid in which network resources (e.g., switch ports and lambdas) can be dynamically scheduled and allocated just like other distributed end resources (compute, storage, scientific instruments, etc.). Such an infrastructure provides revolutionary communication capability because geographically distributed resources can be tightly coupled with dedicated, high-speed optical connections on-demand. In effect, they can be treated as though in the same machine room. This enables a range of innovative distributed applications involving large data objects and collections, large computations and real-time remote data access, only possible with 10's to 100's of Gbps and guaranteed quality of network service.

1.2 The Problem

Although Lambda-Grids provide dramatic opportunities for new computation, communication and collaboration capabilities, building high-performance applications that efficiently exploit resources in such infrastructures is extremely difficult. Significant questions include: how to simplify application use of configurable optical networks and shared, distributed resources; how to select appropriate resources for applications; and how limited network information affects application communication performance and resource utilization.

1.2.1 How to Simplify the Development of High-Performance Applications in Lambda-Grids

When compared to either sequential or parallel programming, the difficulties in developing applications which efficiently exploit configurable optical networks and grid resources are daunting. In particular, applications (or consequently application programmers)

must contend with the complexity of dynamic and unreliable resource environments. End resources (e.g., compute, storage and visualization) are heterogeneous, varying in type, interface, availability and runtime performance. They are drawn from a range of distributed resource providers that represent distinct administrative domains and may impose diverse security and sharing policies on the use of these resources. In the face of these challenges, applications want to achieve high capability and reliable performance, and even tolerate asynchronous changes in resource availability and runtime behaviors.

In addition, applications must deal with the complexity of configurable optical networks in Lambda-Grids. First, utilizing configurable networks requires the applications to understand details of the underlying telecommunication infrastructures, including low-level communication resources (e.g., optical switches, links and lambdas), and how to configure and compose them into desired end-to-end network connections. Second, when a network is partitioned into domains (i.e., multiple distinct service providers), establishing an application across networks requires management of multi-domain optical routing and signaling. Third, delivering the performance of high-speed, long distance connections requires the use of novel, exotic transport protocols. These protocols, including Group Transport Protocols (GTP) [23], UDP-based Data Transport (UDT) [24] and Composite Endpoint Protocol (CEP) [25], are a research activity in and of themselves.

In summary, enabling simple development of high-performance distributed applications in Lambda-Grids requires new system abstractions that hide the complexity of configurable networks and wide-area resource sharing, while exposing novel communication capabilities in a convenient fashion.

1.2.2 How to Select Appropriate Resources for Applications

A significant challenge in achieving high application performance and efficient resource usage in Lambda-Grids is selecting appropriate sets of resources for individual applications. In these environments, we need to match application components (e.g., computation tasks) with suitable end resources (e.g., compute, storage and visualization), as well as select communication resources (e.g., optical switches, ports and links) and compose them into satisfying network connections. Application performance is highly dependent on the quality of end resources and network connections that host its computation and communication. For example, the performance of compute-intensive applications, such as scientific and engineering simulation, depends on the CPU speed, physical memory size and disk space of compute resources. On the other hand, the performance of interactive distributed applications, such as collaborative data visualization, is dependent on the throughput and latency of network connections. With the high heterogeneity and vast number of available resources, applications need to identify and select high-quality resources quickly (within a few minutes) and use them to achieve high performance and low turnaround time.

When thousands of applications take part in the collective sharing of resources, it's necessary to maximize overall system throughput and resource utilization by conserving the use of scarce resources. In Lambda-Grids end computing and storage resources are plentiful, but a core physical network may contain bottleneck links. For example, the Level3 international optical network includes Points-of-Presence (PoPs) in major cities in US and Europe. While these PoPs are highly connected, the bottlenecks are the links crossing the two continents. If there are a large number of applications requesting network paths across the bottleneck links (e.g., allocate compute and storage resources in different continents), they cannot simultaneously run. This could lead to low resource utilization and system throughput.

Therefore, one of the key objectives in the resource selection task is to find a solution with the minimal total distance of optical paths allocated for each application.

1.2.3 How Available Network Information Affect Application Performance and Resource Efficiency

While configurable optical networks are gaining popularity, a significant challenge is controlled network information sharing. Network information (including details of service providers' internal network topologies, link capacity and usage) is essential for effective path computation for grid applications and enables efficient resource usage and high application communication performance. However, with a wide-area network composed of multiple independently managed sub-networks (or domains), a critical tension is between service providers who are business competitors and not willing to share their internal network information to protect their security and competitive positions. This poses key challenges for intelligent network information sharing that must not only maintain competitive advantages of individual service providers, but also enables efficient end-resource and network path selection for distributed applications. Towards this goal, fundamental questions include: what basic types of network information that might be shared between service providers and how individual information factors affect applications' and service providers' ability to utilize resources.

1.3 Previous Work

While Lambda-Grids provide intriguing raw hardware capabilities, the middleware must harness resources into a form usable by applications to achieve high execution performance and enable efficient resource usage. These are difficult goals because the underlying software and hardware infrastructures have daunting complexity and applications

exhibit increasingly complicated and competing requirements for resources. Prior research projects on middleware for Grids [26-30] and wide-area configurable optical networks [17, 18, 31, 32] are far from solutions for application developers with little knowledge of configurable optical networks and Grid environments. In particular, all these systems manage either network or end resources separately, assuming relative simple models for the other domain. Specifically, existing Grid middleware systems [26-30] assume fixed network connectivity (i.e., the Internet) between distributed resources and control only end resources (compute, storage, visualization, etc.). They manage these end resources for high efficiency and performance, but lack the ability to plan and dynamically configure optical networks. On the other hand, optical network services [17, 18, 31, 32] can manage and configure dedicated optical circuits on-demand, but assume publicly accessible end resources. Therefore, they cannot grant users access to end resources across domains. To obtain coordinated use of both resource types, applications must interact with multiple distinct middleware services. Because these services do not coordinate network and end resource selection/allocation, the result is limited application capabilities (e.g., no real-time guarantee and sub-optimal performance) and inefficient use of resources.

Figure 1-1 demonstrates how an application obtains use of compute and network resources in Lambda-Grids with existing middleware. In this example, the application requires two compute clusters and a private optical connection between them. Because compute and network resources are independently managed by Grid middleware and optical network services, the application needs to issue separate requests for the allocation of these resources. Specifically, the application first contacts Grid middleware for allocation of compute resources and later negotiates with optical network services (which represent distinct network service providers) for configuration of a private optical circuit. Although feasible, this approach could lead to two significant problems. First, it causes inefficient use of network

resources. In the former step compute resources are chosen with limited information about available connections; therefore, the allocated resources can be widely distributed and it requires a long optical circuit path (i.e., multiple switch ports and links to be allocated) to realize the required connectivity. Second, in the presence of resource contention, the required network connectivity may not be possible since the longer the optical circuit to be configured, the higher chance some of its resource components (e.g., switch ports and lambdas) are already occupied by other applications.

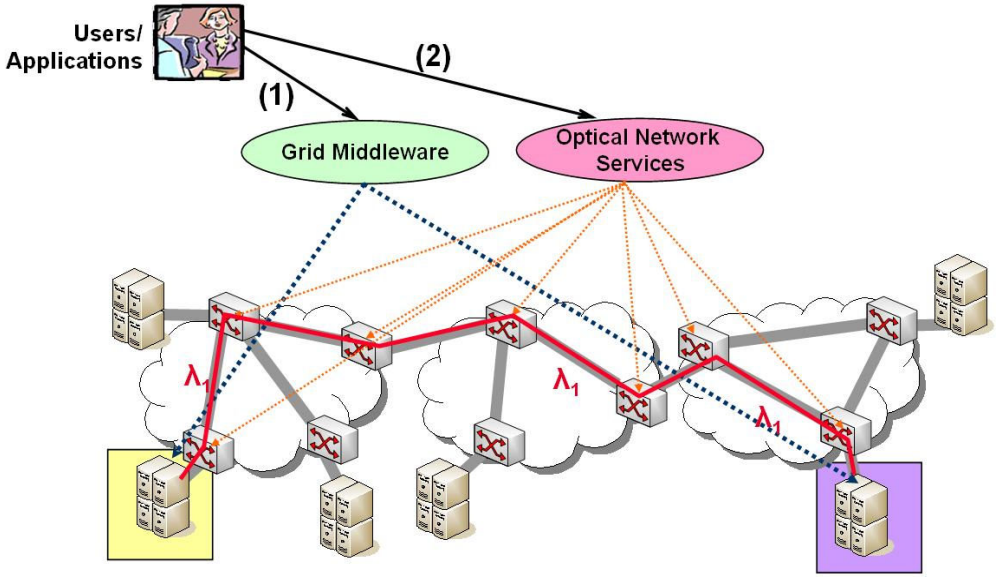


Figure 1-1: Separate management of network and Grid resources. An application sequentially contacts Grid middleware and network services for resource allocation and private network configuration.

1.4 Thesis Statement

Our research investigates an integrated model for managing configurable optical networks and wide-area resource sharing to enable convenient and efficient development of high-performance distributed applications in Lambda-Grids. Our approach uses system abstractions that separate the configuration of resources from the application programming

and execution. In this model, an application (or user) describes its requirements for communication and end resources, and the integrated middleware service is responsible for matching them with an appropriate set of distributed resources and private networks. The selected resources are then automatically configured, reserved and managed; the application can make general use of them as a private distributed presence. Such combined acquisition of both network and end resources enables coordinated resource management improving resource efficiencies and application capabilities.

My thesis is stated as follows:

Guaranteed, high application performance and efficient resource usage can be achieved simultaneously in Lambda-Grids by integrated selection of end resources (e.g., computers, storages and visualization) and network resources.

1.5 Approach

To investigate our thesis, we develop the Distributed Virtual Computer (DVC), an integrated resource management architecture that enables an application to conveniently describe and acquire a combined set of private networks and end resources. The architecture provides a resource specification language in which the application can describe its communication and end resource needs in a general form. Then, the DVC implementation uses this information to drive resource selection and network configuration optimization.

In order to study how to realize application resource requirements effectively, we formulate the resource selection problem, and then explore several implementation approaches. We design and evaluate novel combined resource selection algorithms which use heuristics based on simulated annealing and top-down hierarchical selection. We compare

them to several alternative separate selection approaches of network and end resources via simulation, exploring a range of realistic application models and Lambda-Grid resource configurations. Specifically, we evaluate each algorithm's result quality, cost, and scalability as a function of application request complexity and resource configuration size. Our results show that the combined approaches produce good results for both application performance and resource efficiency, significantly better results than those of separate selection. Additionally, these results are obtained with computational effort low enough that the algorithms could be used online for realistic large-scale Lambda-Grids. This proves our thesis.

Further, we investigate the network information sharing problem. We characterize the information that might be shared by network service providers and study the impact of the available information on the quality of resource selection results in terms of application performance and resource efficiency via simulation. Our experiments use a range of realistic metropolitan, national, and global network topologies derived from Internet Service Providers (ISPs). Our results clearly identify which information factors are important for applications to effectively select network paths and encourage collaboration between service providers to promote overall network efficiency and productivity.

Based on the results from the previous studies, we design and implement the DVC architecture as a system software prototype. We use it to demonstrate the feasibility and effectiveness of the integrated resource management approach with deeper evaluation based on real use with scientific applications and optical networks. The prototype is evaluated in enabling collaborative visualization environments for geosciences on the OptIPuter's Lambda-Grid testbed [15]. This shows that guaranteed, high application performance can be achieved practically.

1.6 Contributions

The primary contribution of this dissertation is an integrated resource management architecture and combined resource selection algorithms that enable high application performance and resource efficiency in Lambda-Grids. Specific contributions are summarized below:

1. **Definition of the Distributed Virtual Computer (DVC), a novel approach for managing network and end resource sharing for Lambda-Grids.** The DVC allows an application to conveniently describe and acquire a set of distributed resources (compute, storage and network) and use them as a private distributed presence. The combined resource acquisition provides opportunities for integrated resource selection allowing for high application performance and resource efficiency to be simultaneously achieved. In addition, the DVC's integrated abstraction simplifies application management of communication and end resources in complex Lambda-Grid environments.
2. **Definition of a resource specification language which describes application resource requirements,** including traditional end resource specification and explicit high-level description of the needed communication resources. Such an expression enables an application to expose unique communication capabilities of Lambda-Grids and enables network service providers to manage their resource utilization for high efficiency.
3. **Design, implementation and evaluation of novel combined resource selection algorithms.** We formulate the resource selection problem, explore several approaches, and evaluate them via simulation. Our results demonstrate the best performance is achieved by techniques which combine the selection of communication and distributed end resources. We present two combined selection algorithms based on simulated annealing and top-down hierarchical selection. Using simulation, both algorithms are

shown to produce good results for both resource quality and efficiency. Further, the algorithm that uses top-down hierarchical technique not only achieves high-quality results, but also scales well with both application request complexity and Lambda-Grid size. This enables an online use where resources can be automatically selected, allocated and configured on-demand in minutes for large-scale scientific applications which typically run for hours or days.

4. **Characterize the impact of network information models on application performance and resource efficiency.** We study the network information sharing problem in a network composed of competing service providers. We characterize the information that might be shared and define a spectrum of network information models. To evaluate the impact of the proposed models, we use simulation across a range of real providers' network topologies. Our results suggest that network providers should make their internal network information available (completely or partially) as it can significantly improve application performance and resource efficiency.
5. **Development of the DVC system software prototype.** We design and implement the DVC system software prototype to demonstrate the feasibility and benefits of the DVC architecture. Key components of this prototype include integrated abstractions for simple application development, combined resource selection for high application capability and resource efficiency, as well as virtual resource names and unified communication interfaces for high application portability and flexibility.
6. **Demonstration of the DVC system software prototype with real scientific applications.** At the iGrid2005 conference, we demonstrate the benefits and capability of the DVC prototype in enabling collaborative and remote data visualization for geosciences. The demonstration shows scientific collaborations can be conveniently, dynamically, and optimally constructed with our software. Such capabilities enable groups

of scientists from separate institutions to interactively analyze the large datasets in real-time. The construction of such applications (without manual configuration by IT administrators) is not possible without our software.

1.7 Dissertation Roadmap

The remainder of this work follows this outline. In Chapter 2, we present the requisite background in Grid computing and configurable optical networks, and discuss related work. Chapter 3 describes the specific context of this work, defines the problem, and presents our thesis statement. In Chapter 4, we describe the design and implementation of the DVC coordinated resource management architecture. We present the resource specification language that describes application resource requirements, virtual resource naming and communication abstractions, and combined resource selection algorithms. Chapter 5 evaluates different resource selection approaches via simulation across a range of realistic application and resource configuration models. In Chapter 6, we evaluate the impact of network information models on the quality of resource selection. Chapter 7 assesses the DVC system software prototype in enabling collaborative visualization environments for earth sciences. Finally, we summarize the dissertation and discuss future research directions in Chapter 8.

1.8 Acknowledgement

Chapter 1, in part, is published as “Distributed Virtual Computer (DVC): Simplifying the Development of High Performance Grid Applications” by Nut Taesombut and Andrew A. Chien in the proceedings of the Workshop on Grids and Advanced Networks (GAN’04), April 2004. The dissertation author was the primary researcher and co-author of this paper.

Chapter 2. Background and Related Work

In this chapter, we present the background and most relevant work in the field. Section 2.1 provides general assumptions on network and end resource hardware for Lambda-Grids. Section 2.2 is an introduction to current approaches for Grid middleware, while Section 2.3 surveys a range of research efforts in enabling dynamic network provisioning. Section 2.4 discusses current approaches for wide-area resource selection. Section 2.5 surveys existing specification languages describing application resource needs in Grids and other wide-area distributed systems.

2.1 Lambda-Grid Architecture

"Grid" computing [9] has emerged as a set of new technologies and infrastructures supporting coordinated resource sharing in dynamic, heterogeneous, and multi-institutional distributed systems. Grid computing has become increasingly important for advanced scientific and engineering applications [1-8] that require large-scale aggregations of distributed computing resources, data objects, and scientific tools to solve complex problems. In this environment, IT administrators of different organizations establish a Virtual Organization (VO) [10], a set of relationships and sharing policies to grant users access to resources across traditional organizational boundaries. These resources can be allocated on-demand and used to achieve high computing, storage and collaboration capabilities. To interconnect remote computing and storage resources, traditional Grid systems have been built over IP packet-switched networks (i.e., Internet) supporting only best-effort service.

Recent advances in optical networking technologies are rapidly changing the current model of wide-area communication, moving from a best-effort, network-constrained world into a deterministic-performance, network-rich world. Dense Wavelength Division Multiplexing (DWDM) has emerged as an efficient technique that allows a single fiber to

carry multiple wavelengths (lambdas), and increases available network capability to several terabits per second. Furthermore, maturing middleware in the network control plane [17, 31, 32] is enabling the ability to dynamically configure and dedicate these high-speed optical circuits to applications on-demand. Such private connections have several key advantages over shared, best-effort Internet connections. These include security and controllable performance properties (guaranteed bandwidth and bounded communication delay and jitter).

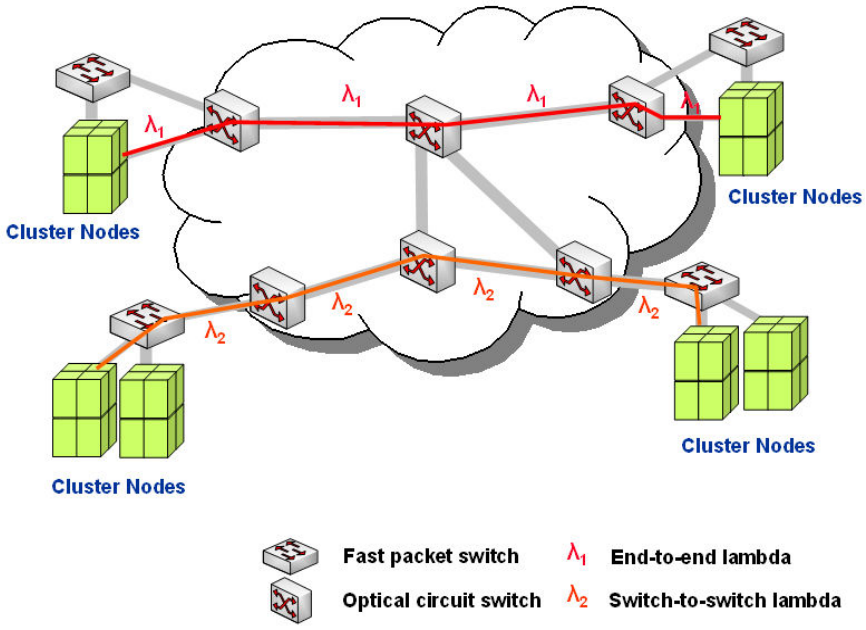


Figure 2-1: Physical architecture of a Lambda-Grid

A Lambda-Grid [22] is a collection of geographically separated end resources (computing, storage, visualization, etc.) that can be securely shared through a VO and tightly interconnected with dedicated optical connections on-demand. Figure 2-1 illustrates the physical architecture of such an infrastructure. Due to the falling price of commodity hardware and the increasing popularity of cluster configurations [33], the end resources are typically organized into clusters. Each cluster contains a collection of homogeneous resources locally managed and tightly coupled via fast Ethernet (packet) switches. The core optical network is

composed of optical switches interconnected by DWDM optical links. Each end resource has one or more optical interfaces paired to one optical switch; these interfaces may be connected via direct links or through fast border packet switches. There are two types of optical circuits (lambdas) that can be configured:

- **End-to-end dedicated lambdas.** These directly connect pairs of end resources with private connections. This approach guarantees quality of network service, but requires each end resource to have a direct optical interface to the core network.
- **Switch-to-switch lambdas.** One or both ends of these lambdas may terminate at a shared border switch; this allows efficient sharing of connections by a set of end resources. Private connections and quality of service can be achieved via VLANs – only end resources participating in the connections are assigned private IP subnet addresses.

In a wide-area Lambda-Grid system, the core network can be partitioned into sub-networks that provide autonomous administrative domains for distinct network service providers. These network providers manage their own communication resources and peer with others to exchange traffic and promote overall network productivity. Configuring optical circuits to interconnect remote end resources may require a distributed network control plane managing interdomain optical routing and signaling.

2.2 Grid Middleware

Grid middleware combines a set of security, resource management, data management, information, monitoring and other services required to efficiently operate a multi-organizational, shared resource environment. Development of grid middleware has received considerable attentions; the notable ones include Globus [26], Condor-G [28], EEGE gLite [29], GridLab [30], and Virtual Grid [27].

The Globus system [26] is a grid middleware realizing the VO concept and providing fundamental grid services for security [34], resource discovery [35], data communication [36], and remote resource access [37]. Globus employs a layered architecture where applications or high-level grid services can be built from lower-level core services. Its strives to provide a simple grid computing environment, defining standard resource configuration, negotiation and communication protocols and presenting uniforms interfaces to applications. For instance, WS-RF [38] is a product of the Globus effort to make Grid resources uniformly accessible through existing web service technologies. Further, the system supports soft QoS guarantee through resource reservation [39]. Unlike the DVC, the Globus system neither manages configurable network resources nor optimizes resource configurations for applications.

The Condor-G system [28] manages compute-intensive jobs for high throughput in a multi-institutional, shared resource environment. It combines Condor's computation management schemes [40] with Globus's interdomain resource management protocols [26] in harnessing idle grid resources and supporting hosting environments for remote job execution. It works in this manner: Users submit jobs to be executed. Condor-G allocates the appropriate resources, initiates and manages computations, and informs of completion or failure. Condor jobs are also automatically checkpointed and migrated between idle machines to ensure correct completion. In contrast to the DVC, the Condor-G system cannot allocate optical resources and guarantee quality of network service for applications.

gLite [29] is the grid middleware of the EU-funded EGEE (Enabling Grids for E-sciencE) project. It leverages existing middleware (including Globus [26], Condor [40] and LCG [41]) to provide high-level grid services that are more efficient and reliable. Key services include workload management for scheduling computational tasks, data management for managing distributed files, and information management for monitoring, collecting, and retrieving grid information. It also provides GridFTP [36] and gLiteIO [42] for high-

performance data transfer and access capabilities. The gLite middleware follows a service-oriented architecture providing compliance with other grid services. Unlike DVC applications, those in gLite cannot exploit dedicated optical circuits to achieve high performance and quality of service guarantee.

The GridLab project [30] aims for a set of high-level grid services and interfaces simplifying the development of grid applications. Key services include grid resource information, brokering, monitoring, visualization and data management. GridLab Resource Management System (GRMS) [43] is a meta-scheduling system that abstracts low-level grid resource management complexity, while including features for load-balancing among clusters, replica management, remote data access and application migration. Grid application developers use these services through a Grid Application Toolkit (GAT) [44]. Unlike DVC applications, those in GridLab need to deal with the complexity of configurable optical networks, including dynamic network configuration and heterogeneous addresses.

The Virtual Grid runtime system [27] enables easy and efficient development of grid applications. It provides a simple abstraction of the grid environment and an integrated set of resource management services allowing applications to acquire high-quality resources quickly and adapt to asynchronous changes in resource conditions and application requirements. This system has a unique “slot” abstraction allowing applications to specify, discover, and allocate resources across time. This improves both scheduling quality and efficient resource use. In contrast to the DVC, the Virtual Grid system doesn’t select and allocate optical network resources for applications.

The key limitation of these systems [26-30] is that they assume the traditional best-effort Internet model and manage only end computing and storage resources. Therefore, they lack the ability to control and expose novel communication capabilities of Lambda-Grids, including dynamic network configuration, high-speed communication, and optical multicast

[45]. Furthermore, most systems focus on the "configuration" and "negotiation" aspects of grid resource management without optimizing resource and network configurations for applications.

2.3 Dynamic Network Provisioning Systems

Many research efforts explore the ability to dynamically configure connections (lambdas) in wide-area, optical circuit-switched networks. Dynamic lambda provisioning not only enables efficient use of communication resources, but also allows applications to flexibly use and modify private connections according to need. There are at least four extant approaches for configurable optical networks.

CHEETAH [18] is a networking framework allowing applications to obtain dedicated, end-to-end optical connections on a dynamic call-by-call basis. Applications submit file transfer requests similar to FTP services (i.e., two end hosts, bandwidth and duration) and CHEETAH optimizes the scheduling, configuration, and use of network resources. The current implementation employs GMPLS control plane [46] to realize dynamic provisioning of optical circuits. The created circuits are held only as long as necessary to complete the transfers. CHEETAH addresses technical issues such as high-speed communication [47] and optical circuit scheduling [48]. However, there are several key remaining issues. These include interdomain routing and signaling, as well as security and heterogeneous network management.

UCLP [32] is a user-controlled lightpath provisioning service architecture that allows applications to create end-to-end optical connections across multiple domains. It models optical resources as lightpath objects (LPOs) that can be described, advertised and discovered through web service technologies. Network providers first create short LPOs (pre-established direct or single-domain lightpaths) and deposit them into a LPO registry. To realize their

communication needs, applications discover, select and compose these short LPOs into composite LPOs or end-to-end connections.

DRAGON [17] is a networking solution that manages dynamic provisioning of deterministic optical paths across multi-domain, heterogeneous networks. Primarily built over GMPLS-based optical networks, it incorporates advanced features for authentication, authorization, accounting (AAA) [49], and resource scheduling. A key component is the Network Aware Resource Broker (NARB) [50] that computes feasible network topologies and instantiates the required connectivity according to need. Applications specify their communication requirements with the Application Specific Topology Description Language (ASTDL) that can express complex topologies (e.g., involving more than two end hosts).

Photonic Interdomain Negotiator (PIN) [31] is a distributed control plane architecture providing dynamic provisioning of end-to-end optical paths across heterogeneous network domains. The architecture is composed of a collection of distributed PIN agents located in different domains that manage interdomain optical routing and signaling. During the dynamic optical path setup, PIN agents along the path translate interdomain signaling messages into corresponding intra-domain signaling messages and pass them to the local control planes. The current implementation employs the Photonic Domain Controller (PDC) service [51] to dynamically establish optical paths within each domain.

These four systems can configure underlying optical connections though presenting diverse interfaces to describe application communication needs. Specifically, UCLP, PIN/PDC and CHEETAH can form a network path between two endpoints per call, while the DRAGON approach can express and instantiate a network configuration composed of multiple paths. Here, network service providers see varying degrees of flexibility for efficient resource management. However, all these systems assume known (or publicly accessible) end points and manage only communication resources. Hence, they cannot optimize application resource

configurations that span both network and end resources (e.g., computing and storage) and cannot coordinate their allocation.

The current DVC system prototype exploits PIN/PDC for dynamic configuration of dedicated optical circuits, but is not limited to it. The DVC is a high-level middleware service that can interface with other dynamic network provisioning systems (including CHEETAH, DRAGON, UCLP, etc.).

2.4 Resource Selection in Wide-Area Systems

Research in wide-area resource selection in grid communities has been widespread. Although Grid computing provides the ability to share and use resources across domains, the distributed ownership of resources leads to the problem of heterogeneous resource usage policies. To run an application, the user needs to find a set of appropriate resources not only matching the application requirements, but also satisfying the imposed use policies of service providers. Consequently, many grid systems formulate the resource selection problem as a "matchmaking" process.

As part of Condor [40], the first matchmaking system [52] was proposed for symmetric, bilateral matching of a single application component with a single resource. When multiple matches are found, resources are ranked to find the one that can produce better performance. However, the system selects a single resource and implements a simple exhaustive search algorithm, thereby limiting its usability and scalability. In Gangmatching [53], the matchmaking framework was extended to support co-selection of heterogeneous resources. The system implements a backtracking search algorithm and uses an indexing scheme which improves the efficiency and scalability of resource selection. Redline [54] also presents a resource selection framework based on symmetric matching, but it reinterprets the selection task as a constraint satisfaction problem and applies a range of constraint-solving

techniques [55]. Unlike our research, neither Gangmatching nor Redline optimizes optical network configurations for applications.

Tangmunarunkit et al. [56] presented an ontology-based resource-matching framework for the Gird. Unlike the aforementioned work, this system doesn't require the symmetric specifications of application requirements and resources. The function is carried out based on rules using domain background knowledge instead of exact syntactical matching. The ontology-based matching system employs a deductive database engine [57] to solve the resource selection problem.

RGIS [58] and R-GMA [59] employ a relational data model to build grid information services that discover and select end resources in response to users' queries. They develop different search heuristics, including non-deterministic, approximate and scoped queries, that allow users to tradeoff between the execution time and the number of results. Our work can benefit from RGIS and R-GMA in querying grid resource availability. In contrast to these systems, our approach combines the selection of both communication and distributed end resources.

Kee et al. [27] introduced the classification for different types of resource aggregates with good or poor shared network connectivity (including 'Cluster', 'TightBag' and 'LooseBag') and built a relational database storing information and selecting end resources based on this classification. In addition to a structured resource database, the system applies a range of simplifying request reduction and query synthesis techniques for efficient and high-quality resource selection. Acceptable resource candidates are ranked based on a user-defined ranking function, and the top candidates are chosen to optimize application performance. In contrast to this system, our approach considers configurable networks and integrates the selection of communication and end resources for both high application performance and network efficiency.

SWORD [60] is a wide-area resource discovery service that selects resources to host applications in large-scale distributed environments. SWORD's design focuses on service robustness and scalability, so it is primarily built over a distributed query processing infrastructure. SWORD supports multi-attribute, range queries and exploits a distributed hash table (DHT) to efficiently store and retrieve resource information from distributed nodes. It allows the user to specify a penalty function on desired resource attributes and provides an optimizer selecting a set of resources with the minimum penalty cost. SWORD's optimizer implements heuristics that first rank candidates by end-resource quality and then select resource configurations (among the top candidates) with good connectivity. In contrast, our approach simultaneously selects communication and end resources for better network efficiency.

Huang and Steenkiste [61] presented an architectural framework for dynamic service composition in a wide-area system. It enables service developers to describe how to compose specific services from distributed end resources and provides a generic synthesizer for optimizing the service configurations. Different resource selection algorithms were proposed to implement the synthesizer, including simulated annealing, simulated annealing with local search, and exhaustive search. These techniques represent the tradeoff between the selection cost and quality. Our approach also employs the simulated annealing technique, but combines the selection of network and distributed end resources.

Emulab's assign [62] and netEMBED [63] formulate the resource selection task as a network-embedding problem. Emulab's assign uses simulated annealing and genetic algorithms for mapping application components onto a small, shared network. The goal here is to maximize resource efficiency. On the other hand, netEMBED employs different search heuristics (including constraint filtering, random walk and lazy neighbor search) to find a valid mapping in a large-scale distributed resource infrastructure. Unlike our work, both

Emulab's assign and netEMBED assume shared network connectivity and don't optimize composed optical connections for applications.

All of the aforementioned systems [27, 52-54, 56, 58, 59-63] assume a fixed connectivity model between end resources and evaluate application communication needs against network performance measurements. In Lambda-Grids, network connections are established on-demand and it is possible to plan and control their capabilities. Network configurability is computationally harder and adds complex planning configurations to end resource selection.

2.5 Resource Specification Languages

A resource specification language describes application requirements and preference for resources. The design of the language is critical; a good one should be expressive to support a wide range of applications and allow them to drive the selection of good-quality resources that can make a major performance difference.

```
&(executable= '/home/user1/bin/a.out')
(directory = '/home/user1')
(environment = (DATADIR '/home/user1/data'))
(|(&(count=4)(disk>200))(&(count=8)(disk>100)))
```

Figure 2-2: A sample RSL specification that describes the resource needs of a job

The Globus resource specification language (RSL) [64] provides a declarative view of resource requests exchanged between components of the Globus Resource Management architecture. The RSL also provides a way for applications to describe the resource requirements of submitted jobs to the Grid. The core RSL syntax is a hierarchical structure of relations, where each relation (<*attribute,value*>) associates an attribute name with a value. Figure 2-2 illustrates an example RSL specification that describes a job's request for four

compute nodes with at least 200 GB of disk space or 8 nodes with at least 100 GB of disk space.

```
[
  type = “request”;
  user = “UserA”;
  org = “UCSD-CSE”;
  requirement = cpuSpeed > 2.4 && memory > 1024;
  rank = other.cpuspeed*1000 + other.memory
]
```

(a)

```
[
  type = “resource”;
  cpuSpeed = 3.2;
  memory = 1024;
  diskSpace = 250;
  user-list = {“UserA”, “UserB”, “UserC”};
  requirement = IsMember(other.user, user-list)
]
```

(b)

Figure 2-3: Example ClassAd specifications: (a) an application request; and (b) a resource description

The Condor ClassAd language [65] allows service providers and users to describe the capabilities and requirements of resources to be used in a matchmaking process. A ClassAd specification includes: 1) properties of this ClassAd; 2) constraints that must be satisfied by a single or set of matching ClassAd(s); and 3) a ranking function that describes preference on matching ClassAds. A collection of ClassAds match if all their constraints are satisfied. Figure 2-3 (a-b) show two examples of ClassAds respectively describing an application request and a resource.

```
request = [
  user = “UserA”;
  comp ISA SET [CPUSpeed > 2.4; memory > 1024];
  storage ISA [DiskSpace > 300]
  Forall x in comp;
  x.bandwidth[storage.hn] > 30;
  Sum(comp.CPUSpeed) > 6;
]
```

(a)

```
storage1 = [hn = “hostA”; DiskSpace = 400]

computer1 = [CPUSpeed = 3.2; memory = 2048;
  bandwidth = DICTIONARY [{“hostA”, 40},
  {“hostB”,20}]]

computer2 = [CPUSpeed = 3.0; memory = 4096;
  bandwidth = DICTIONARY [{“hostA”, 32},
  {“hostB”,10}]]
```

(b)

Figure 2-4: Example Redline specifications: (a) an application request; and (b) resource descriptions

Redline [66] is a constraint specification language that describes properties or requirements of resources. Unlike other constraint languages, it incorporates new types of constraints and predicates more suitable for specifying a request/resource. Compared to ClassAd and RSL, the constraint language approach allows more expressive specification of application resource needs, including complex resource aggregates and set-related functions. Figure 2-4 (a-b) respectively illustrate a sample request specification and three specifications for resources. The former describes a request for a group of compute machines, a storage resource and a network connection.

```
Group group1
  NumMachines 4
  Required CpuSpeed [1.0, 3.2]
  Preferred CPUSpeed [2.4, 3.2], penalty 100.0
  Required AllPairs BW [10, MAX]
  Preferred AllPairs BW [20, MAX], penalty 10.0

Group group2
  NumMachines 2
  Required FreeDisk [100, MAX]
  Preferred FreeDisk [300, MAX], penalty 80.0
  Required AllPairs BW [10, MAX]
  Preferred AllPairs BW [20, MAX], penalty 10.0

InterGroup
  Required OnePair BW group1 group2 [5, MAX]
```

Figure 2-5: A sample SWORD query describing application resource needs

SWORD [60] provides a query specification language expressing application requirements for resources. Queries in SWORD often contain: 1) a number of equivalent resource groups, each defined by a number of required nodes, a range of acceptable performance attributes, as well as inter- and intra-group connectivity; and 2) penalty functions [67] that guide detailed choices amongst acceptable resources to optimize application

performance. Figure 2-5 illustrates an exemplary SWORD query requesting two resource groups and network connectivity between them.

```
VGrid1 = LooseBag<ClusterNode>[4-8]{ClusterNode =
Cluster<Node>[8-16] {Node = {memory > 1024MB, CPUSpeed > 1GHz}}
Rank(Node) = CPUSpeed
```

Figure 2-6: A sample vgDL resource specification for a loosely-coupled group of compute clusters

The vgDL description language [68] specifies Virtual Grids [27], which provide simplifying resource abstractions for grid applications. A vgDL specification describes resource needs as different hierarchies of resource aggregates (including 'Cluster', 'LooseBag' and 'TightBag'), used by applications to express forms of parallelism and communication optimization. It also specifies a ranking function guiding detailed choices amongst acceptable candidates. Figure 2-6 illustrates an example vgDL specification requesting a loosely-coupled group of compute clusters.

Traditional resource specification languages [60, 64-66, 68, 71] express application communication needs implicitly as end resource attribute constraints (e.g., server1.Bandwidth[server2] > 100 Mbps). This approach is limited to specification of end-to-end connectivity requirements and cannot describe complex communication structures of Lambda-Grids, including sharing properties, multi-path connections and optical multicast [45].

Chapter 3. Thesis Statement

3.1 Context

Emerging large-scale scientific and engineering applications depend on distributed cyber-infrastructures. Example applications include distributed computational steering [2], collaborative data visualization [3], scientific data distribution and sharing [1] and distributed content delivery. Typically, such applications are compute- and communication-intensive, requiring access to massive collections of distributed compute resources and data objects as large as several terabytes. In support, underlying infrastructures must deliver dramatic compute and storage capabilities and high quality network services which include extreme bandwidth (tens or even hundreds of gigabits per second) and controllable jitter and latency.

With the emergence of Lambda-Grids, large-scale aggregations of compute clusters and petabyte data stores with high-speed and predictable network performance to support scientific applications have become possible. Such infrastructures enable applications to exploit dedicated optical circuits to tightly interconnect geographically dispersed resources across organizational boundaries. Applications can make use of these resources to achieve high performance, synchronous collaboration, quality of service and real-time guarantee.

While Lambda-Grids provide intriguing raw hardware capabilities, there are significant challenges about how to support the application use of these complex resource environments. For effective utilization, a service model is needed that is both simple to use and delivers key novel capabilities. The infrastructures admit a wide range of possible service models, varying in optimization objective, level of application visibility, and granularity of resource allocation. Our research focuses on the use of a Distributed Virtual Computer (DVC), an integrated application resource abstraction, as a service model. In this model, an application requests and acquires a private collection of resources which combine distributed end

resources (including computers, storages and visualization) with a set of dedicated optical circuits. These resources are then bound into a single virtual domain and transparently managed for high performance and synchronous collaboration. An application in this virtual environment can have direct, secure, high-speed and reliable access to remote resources.

3.2 Problem Definition

To enable the DVC abstractions and the service model outlined above, a number of significant and difficult challenges must be addressed. These include: how to provide a virtual application resource environment, how to represent application resource needs, how to efficiently select resources for high-performance applications, and how limited network information affects application performance and resource utilization.

3.2.1 How to Provide a Virtual Application Resource Environment

Although feasible, building applications that effectively exploit wide-area resource sharing and novel communication capabilities of Lambda-Grids is difficult. Such applications must contend with the complexity of Grid environments, which are distributed, dynamic, heterogeneous and untrusted in terms of resources and networks involved. Grid resources span many administrative domains and vary in type, performance property, availability and naming mechanisms (e.g., full naming, dynamic and internal network address). Furthermore, utilizing network configurability directly requires an understanding of complex telecommunication infrastructures and application management of interdomain optical routing and signaling. Without simplified interfaces to such computing environments, developing applications to manage resource dynamics, heterogeneity and network configurability is impractical.

Additionally, delivering the performance of high-speed, long distance connections require the use of novel, exotic transport protocols. Lambda-Grids allow dynamic construction

of private, high-speed wide-area networks. Many advanced transport protocols [23-25, 69, 70] are being developed for delivering novel communication capabilities in such networks. These protocols are optimized for certain communication paradigms (e.g., point-to-point vs. collective, and streaming vs. reliable), and applications can exploit a mixture of these protocols to optimize their data flows. However, utilizing different protocols with varied interfaces complicates application programming.

Our objective is to develop programming abstractions and tools which allow for the construction and deployment of applications in a convenient way similar to a local private distributed computing environment. Toward that goal, there are open questions about how to provide a uniform access to heterogeneous resources, how to handle dynamic resource changes, how to provide a uniform communication interface, and how to abstract multi-domain security.

3.2.2 How to Represent Application Resource Requirements

Another significant challenge is the definition of a specification language that describes application resource requirements, including distributed end resources and private optical networks. The language's design is critical because it allows applications to share specific knowledge of their resource needs and drive resource selection and network configuration optimization. Ideally, it must be expressive enough to describe unique communication structures of Lambda-Grids (e.g., connection sharing and optical multicast), while supporting simple specifications for inexperienced users such as scientists. In addition, the language should allow expression of abstract resources and network topologies providing the underlying resource planning services with sufficient flexibility to optimize resource configurations for the application and enable efficient resource use.

To date, most resource specification languages for Grids [60, 64-66, 68, 71] describe application communication requirements implicitly as end resource property constraints. This is a limited expression of end-to-end network connectivity and cannot describe unique communication structures in Lambda-Grids, such as sharing properties and photonic multicast.

3.2.3 How to Select Resources for Applications

The selection of appropriate resources for individual applications is an important challenge in achieving high application capabilities and resource efficiency. Application performance is dictated by a range of factors; these depend on the application's characteristics. For example, communication latency and jitter are critical for real-time applications, and compute clusters' CPU speed and physical memory are critical for high-performance computing applications. With the increasing number of available resources in Lambda-Grids (as many as hundreds of thousands), it's necessary to identify and select good resources quickly for applications to achieve high-performance or real-time execution. In addition (with many users joining in the collective sharing of resources), it is also important to conserve the use of scarce resources, such as bottleneck network links, in order to maximize the system throughout.

Our resource selection problem has many similarities to resource selection problems for Grids [27, 53, 60] and network embedding problems [62, 63]. However, network configurability presents unique challenges, adding the complexity of planning configurations to that of end resource selection. This type of network planning is more difficult than end-resource selection as it involves the optimization of composed communication resources.

3.2.4 How Available Network Information Affects Application Performance and Resource Utilization

With a network comprised of multiple independent Internet Service Providers (ISPs), a key challenge is controlled information sharing not only enabling efficient resource selection, but also maintaining the competitive advantages of individual providers. Due to numerous issues of trust, security and economics, ISPs hide sensitive information about their networks, and the limited information available may affect the quality of resource selection. Consequently, this may impact application communication performance and resource utilization in Lambda-Grids. Significant and open research questions are what types of information matters and how it affects applications' and ISPs' ability to utilize network resources.

3.3 Thesis Statement

Our research investigates an integrated framework for managing configurable optical networks and wide-area resource sharing in Lambda-Grids. Such integrated management enables coordinated resource use and combined resource selection for optimizing configurations for applications. My thesis is stated as follows:

Guaranteed, high application performance and efficient resource usage can be achieved simultaneously in Lambda-Grids by integrated selection of end resources (e.g., computers, storages and visualization) and network resources.

To prove the thesis, we first need to demonstrate the feasibility and advantages of the integrated resource management idea. Our approach develops an integrated resource

management architecture allowing individual applications to describe and acquire a combined set of network and end resources in Lambda-Grids. This architecture provides a simple service model for applications, enables applications to share information about their resource needs, and allows integration of resource and network configuration optimization. Guaranteed application performance is achieved by dedicating use of resources for the entire duration of application execution. Further, the combined resource management enables construction of integrated resource abstractions for applications, including a single security domain, a single namespace, and uniform resource access. These abstractions enable complex collections of highly dynamic and heterogeneous resources to be used like a physically secure, localized LAN/SAN environment.

Combined resource selection is our approach to achieve high application performance and resource efficiency in Lambda-Grids. Such approach enables optimization to span both network and end resources, and simultaneously achieves both goals. While combined selection offers great flexibility of choices, it is difficult computationally and algorithms that guarantee optimal solutions are not practical at resource scale (thousands to millions of resources). Our approach uses heuristics based on simulated annealing and top-down hierarchical selection. Such techniques improve selection time significantly while providing high-quality resource selection without compromise.

To study the impact of limited network information on application performance and resource efficiency, a spectrum of network information models is defined and we assess how the choice of model affects the quality of resource selection. For our evaluation to broadly useful, we consider a range of realistic ISP optical network topologies at metropolitan, national and global scales. This also enables us to study the impact of network topology design on the utility of different network information factors.

As a final proof of our thesis, we build a system software prototype which realizes the integrated resource management architecture. The prototype is used to demonstrate the effectiveness of our integrated resource management approach with greater credibility and deeper evaluation based on real use with scientific applications. This prototype shows that guaranteed, high application performance is achieved in Lambda-Grids.

3.4 Acknowledgement

Chapter 3, in part, is published as "Distributed Virtual Computer (DVC): Simplifying the Development of High Performance Grid Applications" by Nut Taesombut and Andrew A. Chien in the proceedings of the Workshop on Grids and Advanced Networks (GAN'04), April 2004. The dissertation author was the primary researcher and co-author of this paper.

Chapter 4. System Design and Implementation

This chapter presents the design and implementation of the Distributed Virtual Computer (DVC), an integrated middleware service for easy and efficient development of high-performance applications on Lambda-Grids. Key aspects of the DVC include a simple model of use for applications, a resource description language that integrates application resource needs, simple naming and communication interfaces that integrates a wealth of underlying network complexity, as well as combined resource selection for high application capability and resource efficiency.

This chapter is organized as follows. Section 4.1 introduces the Distributed Virtual Computer and its key components. Section 4.2 presents the DVC Integrated Specification Language describing combined application resource needs. Section 4.3 discusses key DVC abstractions that simplify application management of security, resource, naming and communication. Section 4.4 presents two combined resource selection algorithms based on simulated annealing and top-down hierarchical selection. Section 4.5 discusses the implementation of the DVC system prototype. Section 4.6 summarizes the chapter.

4.1 Overview of Distributed Virtual Computer

The Distributed Virtual Computer (DVC) architecture supports coordinated resource management and provides a simple service model for applications. The architecture builds on two insights from our experience in developing distributed applications and services on Lambda-Grids. First, building applications that exploit network configurability and cross-domain resource sharing is extremely difficult. Application developers need to understand details of the underlying software and hardware infrastructures and manage highly dynamic and heterogeneous resources. The selection and configuration process of wide-area distributed resources is tedious and error-prone. Second, supporting synchronized use of network and end

resources requires coordination between network services and Grid resource managers. This is impossible without a well-developed model of use (or service model) with Grid resource and network service providers.

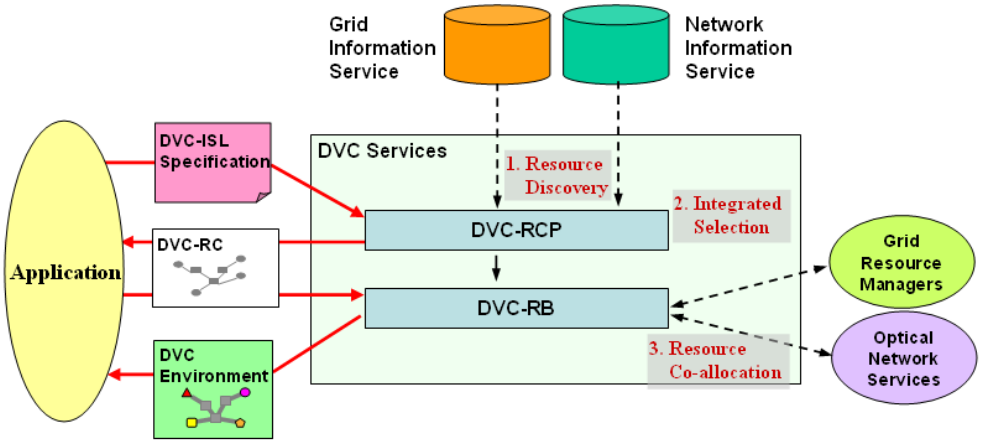


Figure 4-1: DVC integrated resource management architecture

To support easy and efficient development of high-performance applications on Lambda-Grids, we develop the DVC integrated resource management architecture shown in Figure 4-1. Key elements include:

- **DVC-ISL** – an "integrated specification language" that describes application resource requirements, including traditional end resource specification and explicit high-level description of communication resources. The language allows applications to share specific knowledge of their resource needs and drive resource selection and network configuration.
- **DVC-RCP** – a "resource configuration planner" which takes a DVC-ISL specification as input, and (subject to information about currently available resources) returns a resource configuration matching the specification. The DVC-RCP integrates resource selection and

network planning producing the configuration that enables high application capability and resource efficiency.

- **DVC-RC** – a "resource configuration" that is the output of the DVC-RCP and describes a combined set of distributed end resources and optical networks to realize the application resource requirements as given in the DVC-ISL.
- **DVC-RB** – a "resource binder" which takes a DVC-RC as input and negotiates with Grid resource managers and optical network services for co-allocation of the end resources and private networks as described in the DVC-RC.
- **DVC environment** – a middleware-enabled, configurable collection of private end resources and optical networks transparently managed for guaranteed, high performance and quality of service. The DVC environment also provides a simple set of abstractions to simplify application management of naming, security, communication and resources.

Here, an application (or user) can conveniently describe, acquire, and use a private set of distributed end resources and optical networks. Specifically, the application creates a DVC-ISL specification describing its resource requirements, and passes it to the resource configuration planner (DVC-RCP). In response, the DVC-RCP retrieves information about available resources from Grid and network information services and matches the given specification with a resource configuration (DVC-RC) that represents an appropriate set of network and end resources. The DVC-RC is then presented to the application. If not satisfied with the result, the application modifies its specification and repeats the process. Then, the application passes the DVC-RC to the resource binder (DVC-RB) that instantiates it by allocating the corresponding end resources and optical networks. All these resources are then bound into a DVC environment, a simplified computing environment with the complexity of use comparable to a private, local distributed system. Within the environment, the application uses these resources to achieve secure, high-performance, reliable execution. In addition, the

application can configure the DVC environment and modify its configurations according to evolving application requirements and resource conditions. As an example, the application can dynamically add end resources and optical connections into the environment. It can also organize the allocated resources into sub-domains for group management.

The proposed architecture has integrated resource management. This allows for coordination between communication and end resources to achieve high, guaranteed application performance. Furthermore, the combined acquisition of both resource types provides opportunities for integrated resource selection which can improve application capabilities and resource utilization in Lambda-Grids.

4.2 Integrated Specification Language

A key element of the DVC approach is a resource specification language (DVC-ISL), describing application communication and end-resource requirements and allowing their use for driving both resource selection and network configuration. These DVC-ISL specifications are simple, making it easy for high-level users with little knowledge of the underlying resource infrastructures, yet expressive, allowing expression of complex network structures in Lambda-Grids.

The DVC-ISL language builds on the Redline constraint language [66] adding simple extensions for explicit high-level description of network structures (i.e., communication nodes and links are named). Such explicit description enables applications to expose the unique communication capabilities of Lambda-Grids; these include photonic multicast [45] and connection sharing properties. The DVC-ISL differs from traditional description languages for Grids [60, 64-66, 68] where application communication requirements are implicitly specified as end resource attribute constraints (e.g., server1.Bandwidth[server2] > 1Gbps), which is limited to the expression of end-to-end network connectivity.

It should be noted that the set of switches and links in the desired networks do not have to fully specify. Applications simply describe high-level network connectivity constraints (e.g., end-to-end bandwidth and latency) and leave the task of composing network resources to the DVC-RCP. Explicitly specifying network structures only provides a means for applications to share specific knowledge about their communication needs and this allows opportunities for optimizing application performance and resource efficiency.

DVC-ISL-Spec	::= **Identifier** '**=**' '**[**'*StatementList* '**]**'
StatementList	::= *Statement* ['**;**' *Statement*]*
Statement	::= **Identifier** '**ISA**' '**[**' *RL_ConstList* '**]**'
	| **Identifier** '**ISA SET**' '**[**' *RL_ConstList* '**]**'
	| **Identifier** '**ISA CONN**' '**(**' *ReferenceList* '**)**' '**[**' *RL_ConstList* '**]**'
	| **Identifier** '**ISA CNODE**' '**[**' *RL_ConstList* '**]**'
	| *RL_Const*
	| *RankFunc*
ReferenceList	::= *Reference* ['**,**' *Reference*]*
RankFunc	::= ('**Maximize**' | '**Minimize**') '**(**' *RL_ArithExpr* '**)**'

Figure 4-2: The BNF description of the DVC-ISL resource specification language

Figure 4-2 provides a BNF description of the DVC-ISL excluding parts of the original Redline language (*RL_ConstList* and *RL_ArithExpr*). The full BNF description is given in the Appendix. Key features of the DVC-ISL language include.

- Resource constraints and preferences
- Resource aggregates
- Internal communication nodes
- Network connectivity

4.2.1 Resource Constraints and Preferences

To support applications that are sensitive to resource and network performance characteristics, the DVC-ISL can specify constraints on resources with acceptable ranges of their per-node or collective attributes, such as CPU speed, physical memory, disk storage and processor architecture type. Evaluations are run with these constraints against current resource performance status and availability information in a database. These determine which resources are acceptable and can be allocated for the requesting application. Further, to enable applications to guide detailed choices among acceptable candidates that can make major performance difference, the DVC-ISL supports specification of a "ranking function". An arithmetic expression, the ranking function computes and compares the quality of resource candidates. The use of a ranking function informs the resource configuration planner (DVC-RCP) to choose the "best" resource candidates that have either maximum or minimum rank value.

4.2.2 Resource Aggregates

Many scientific and engineering applications require aggregations of resources for large-scale computation, data repositories and high-performance visualization. The required computing nodes and disk storages are often generic and interchangeable; they may share some common attributes such as processor architecture and operating system types. To capture these resource aggregate requirements, the DVC-ISL supports requests for "resource sets" corresponding to collections of homogeneous resources. These resource sets can have different types of network connectivity, depending on the specified communication constraints.

4.2.3 Internal Communication Nodes

Applications are not all alike. Different types require differing transport services. For example, high-performance computing applications need the ability to exchange data among many computing nodes, while collaborative data visualization applications require the ability to multicast information. One possible way to meet these different requirements is to use multiple point-to-point connections. However, this approach is costly and perhaps infeasible in the presence of resource contention. Instead, these requirements can be efficiently realized with the use of special network hardware such as photonic multicast and aggregate switches.

To capture complex network requirements, the DVC-ISL can describe internal communication nodes (or "cnodes") in the network. A cnode allows traffic exchange between its interfaces (or links connecting to it) and may offer different types of transport services. These include:

- **Cross-connect** – the ability to exchange traffic between a fixed pair of its interfaces
- **Exchange** – the ability to exchange data traffic between any pair of its interfaces. This capability allows for traffic aggregation and connection sharing.
- **Multicast** – the ability to optically duplicate an input signal and deliver it to multiple output interfaces.

By default, a cnode corresponds to an exchange switch. When large numbers of end resources are in need of full connections, using exchange switches can reduce the number of optical circuits required (*n* compared to (*n*(n-1))/2* circuits).

4.2.4 Network Connectivity

The DVC-ISL supports three types of network connectivity constraints, including "intra-cluster", "lambda" and "internet".

- **Intra-cluster** – This connectivity constraint is defined against a resource set indicating all the resources must be machines within the same physical cluster. In a cluster environment, a collection of resources are tightly coupled via local high-speed and low-latency Ethernet switches.
- **Lambda** – This constraint specifies that two or more endpoints (end resources or cnodes) are interconnected via private optical circuits. Each circuit is a direct, secure, and congestion-free path between resources. It provides high transport performance and guaranteed quality of service.
- **Internet** – This constraint indicates that two or more endpoints are interconnected via a shared packet-switched network or the Internet. This allows aggregation of traffic from multiple endpoints. However, it may cause congestion and unpredictable performance.

4.2.5 Example DVC-ISL Specifications

Here, we present two DVC-ISL specification examples and highlight their advantages.

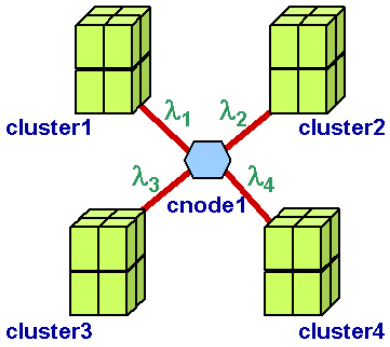


Figure 4-3: An abstract resource configuration for tightly coupled sets of compute clusters

```
(1): ClusterSet =[
(2):    cluster1 ISA SET [CPUSpeed >= 2.4; Memory >= 1024]; Count(cluster1) == 8;
(3):    cluster2 ISA SET [CPUSpeed >= 2.4; Memory >= 1024]; Count(cluster2) == 8;
(4):    cluster3 ISA SET [CPUSpeed >= 2.4; Memory >= 1024]; Count(cluster3) == 8;
(5):    cluster4 ISA SET [CPUSpeed >= 2.4; Memory >= 1024]; Count(cluster4) == 8;
(6):    conn1 ISA CONN (<cluster1>)[type="intra-cluster"; Bandwidth>=1000];
(7):    conn2 ISA CONN (<cluster2>)[type="intra-cluster"; Bandwidth>=1000];
(8):    conn3 ISA CONN (<cluster3>)[type="intra-cluster"; Bandwidth>=1000];
(9):    conn4 ISA CONN (<cluster4>)[type="intra-cluster"; Bandwidth>=1000];
(10):  cnode1 ISA CNODE [Required(exchange)];
(11):  lambda1 ISA CONN (<cluster1>,<cnode1>) [type = "lambda"; Bandwidth >= 1000 ;Latency <20];
(12):  lambda2 ISA CONN (<cluster2>,<cnode1>) [type = "lambda"; Bandwidth >= 1000 ;Latency <20];
(13):  lambda3 ISA CONN (<cluster3>,<cnode1>) [type = "lambda"; Bandwidth >= 1000 ;Latency <20];
(14):  lambda4 ISA CONN (<cluster4>,<cnode1>) [type = "lambda"; Bandwidth >= 1000 ;Latency <20];
(15):  Maximize((Avg(cluster1.CPUSpeed) + Avg(cluster2.CPUSpeed) + Avg(cluster3.CPUSpeed) +
       Avg(cluster4.CPUSpeed))/4)
(16):  ]
```

Figure 4-4: A sample DVC-ISL specification for tightly coupled sets of compute clusters

Many scientific applications [27] require large-scale resource aggregations for complex computation. Large sets of distributed computing resources in Lambda Grids can be assembled from multiple organizations and tightly interconnected with dedicated optical circuits. Figure 4-3 and Figure 4-4 illustrate a sample resource configuration for tightly coupled sets of compute resources and a DVC-ISL specification describing it. Specifically, Lines 2-5 describe four resource sets, each containing eight compute nodes, and specify each node has CPU speed faster than 2.4 GHz with a physical memory size greater than 1024 MB. To define all nodes in each resource set to be machines within the same physical cluster, Lines 6-9 specify the intra-cluster connectivity constraints on them. Lines 10-14 specify optical connectivity between these clusters. Instead of asking for network connectivity between every pair of clusters, the specification describes an exchange switch and optical connectivity from each cluster to this communication node. This approach reduces the number of required optical circuits if a large number of clusters need to be all interconnected. Further, by describing network connectivity between them, the resulting links connecting different clusters can be efficiently shared among communications going from any node in one cluster

to any node in another. This can be implemented by allocating optical circuits terminating at fast border packet switches attached to each cluster. Line 15 defines a ranking function with the optimization objective to maximize the average CPU speed of the four clusters.

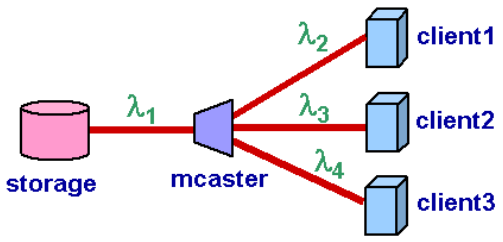


Figure 4-5: An abstract resource configuration for a multicast group

```
(1): MulticastGroup =[
(2):    storage ISA [InSet(DataSet, "Potomac.scene")];
(3):    client1 ISA [Hostname == "grid3.ucsd.edu"];
(4):    client2 ISA [Hostname == "igrid111.cs.uic.edu"];
(5):    client3 ISA [Hostname == "dream.cse.uci.edu"];
(6):    mcaster ISA CNODE [Required(opt-mcast)];
(7):    lambda1 ISA CONN (<storage>,<mcaster>) [type = "lambda"; Bandwidth == 1000 ;Latency <20];
(8):    lambda2 ISA CONN (<mcaster>,<client1>) [type = "lambda"; Bandwidth == 1000 ;Latency <20];
(9):    lambda3 ISA CONN (<mcaster>,<client2>) [type = "lambda"; Bandwidth == 1000 ;Latency <20];
(10):  lambda4 ISA CONN (<mcaster>,<client3>) [type = "lambda"; Bandwidth == 1000 ;Latency <20];
(11):  Minimize(lambda1.Latency + Avg(lambda2.Latency + lambda3.Latency + lambda4.Latency))
(12):  ]
```

Figure 4-6: A sample DVC-ISL specification for a multicast group

Photonic multicast [45] is an efficient communication structure for one-to-many communications. It doesn't require optical-electronic-optical (OEO) conversion and only one transmitter is needed; thereby, it provides high bandwidth, low latency, and efficient resource use. Figure 4-5 and Figure 4-6 show a sample of photonic multicast and a DVC-ISL specification describing it. Specifically, Line 2 specifies a data source, a storage machine that contains a particular dataset. Lines 3-4 specify multicast clients at three known locations. Lines 6-10 describe a multicast connection where a communication node with the 'multicast'

capability and optical circuits from this node to individual clients are defined. Line 11 defines a ranking function with the optimization objective to minimize the average latency between the data source and three clients.

4.3 Application Resource Abstraction

To simplify application use of resources in Lambda-Grids, we develop a DVC environment, an integrated resource abstraction providing applications with simple usage and performance model. The DVC environment provides an abstraction layer insulating applications from the full complexity of building robust and secure applications on highly dynamic and heterogeneous resource environments. Key DVC abstractions include a single security domain, virtual resource names and groups, uniform resource access, and unified communication interfaces. Altogether, the resulting programming complexity is comparable to a private, locally distributed computing environment.

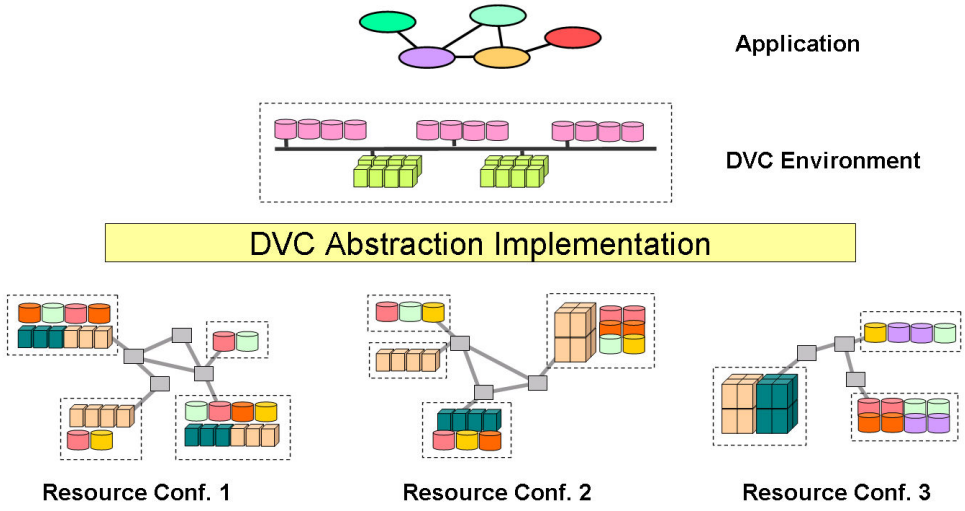


Figure 4-7: DVC resource abstractions enable a simple view of a private local distributed computing environment under a single security domain

Figure 4-7 illustrates a high-level view of a DVC environment (shown in the middle). This environment can be viewed as a collection of resources assembled from several remote sites. These sites may span multiple administrative domains (shown with distinct dotted boxes) and enforce diverse resource management, naming and security policies. The DVC abstractions enable a simple computing environment where the assembled resources are tightly connected via a reliable, private network and controlled under a single administrative domain. For the application to be able to run on different physical resource configurations without modification, the DVC environment provides a private virtual namespace and uniform access to the distributed resources.

- **Single security domain** – each DVC environment is private to a single or groups of users; all logically grouped resources are centrally managed under a single security domain. DVC implementation mechanisms ensure it is secured as a local private distributed environment.
- **Virtual resource names and groups** – to support application flexibility and portability across different physical resource configurations, the DVC environment provides a simple virtual resource namespace (hostnames and IP addresses). Communication among these names is implicitly tied to the dynamically configured networks and allocated resources. Resources can also be identified as groups; this provides for easy group-based management.
- **Uniform resource access** – the DVC environment allows direct, uniform access to distributed resources, masking heterogeneous access policies and mechanisms imposed by distinct resource providers. Resources are allocated and configured prior to application execution time. Then, they can be accessed directly via DVC control channels and simple interfaces.
- **Unified communication interfaces** – To provide uniform access to the advanced transport protocols [23-25, 69, 70] needed to deliver high performance; the DVC environment

provides a unified communication interface. This interface leverages the DVC namespace and groups and provides convenient access and procession of novel capabilities such as photonic multicast.

4.3.1 Single Security Domain

To simplify application management of multi-domain security in Grids, a DVC environment provides a single security domain. Each environment is private to a single or group of users. Within the DVC environment, the user has full control over the allocated network and end resources. Each environment is initiated with a base set of security properties, but the user may set various levels of security and trust amongst resources, including network. Based on the configured security level, the underlying DVC services select and implement appropriate security mechanisms prior to application runtime. The application assumes a secure computing environment during actual execution.

To realize the DVC single security domain abstraction, the DVC relies on and leverages the Globus's Grid Security Infrastructure (GSI) [34] that provides standard security mechanisms for authentication, authorization, and secure remote job invocation. The GSI implements a Virtual Organization (VO), a set of relationships and sharing policies permitting coordinated use of distributed resources across traditional organizational boundaries by a community of users. However, VO's are configured cooperatively among the IT administrators of all participating organizations, so change is difficult and slow. The Community Authorization Service (CAS) [72] is a VO-enabled service eliminating the need for direct interaction between resource providers, thus addressing the flexibility and scalability problems of the VO construction. The DVC environment assumes an existence of VO and CAS, but it is a dynamic application instance oriented structure. A VO can be easily

instantiated within a DVC by a single user, and may come and go dynamically when a single application runs.

To contend with complex security policies required by large-scale collaborative applications such as scientific data distribution and sharing [1], finer-grained security domains can be formed across a subset of resources within the DVC environment. Users may set access control or a trust level for individual resources and security sub-domains. Three security options are available: 1) trusted network and resources; 2) trusted network and untrusted resources; 3) untrusted network and resources. Depending on the option selected, proper security mechanisms like authentication, authorization and encryption are transparently implemented and enforced within the DVC environment.

4.3.2 Virtual resource names and groups

To aid application portability and resource management, the DVC environment provides a virtual private namespace (IP addresses and hostnames) for resources. With the allocation of a new resource into the DVC environment, it is assigned a unique virtual hostname and IP address. An application can explicitly choose meaningful names to simplify organization of resources. The virtual namespace insulates the application from the complexity of heterogeneous resource naming mechanisms (e.g., full naming, dynamic and internal network addresses) imposed by distinct resource providers. It enables the application to run on different physical resource configurations without the need for modification.

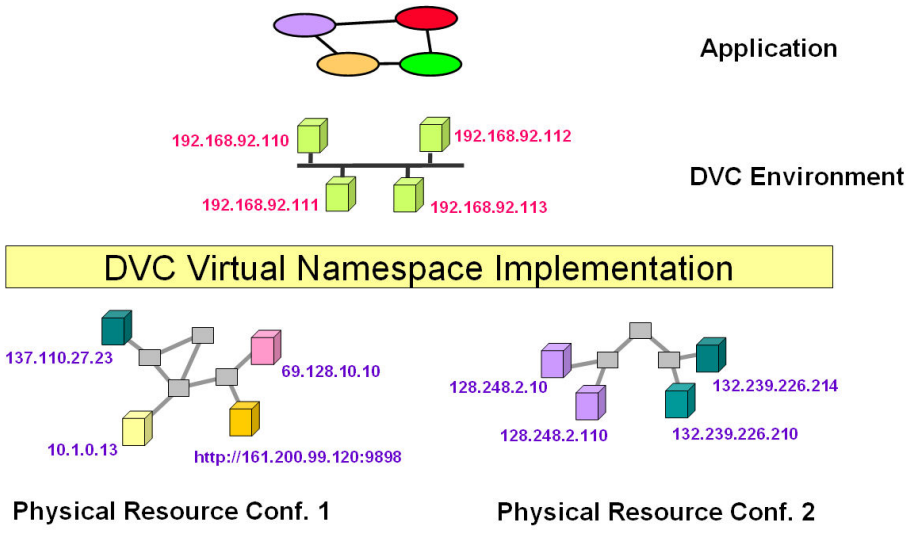


Figure 4-8: The DVC virtual namespace simplifies application management of heterogeneous resource names and aids application portability

Figure 4-8 illustrates a sample DVC environment implemented with different physical resource configurations on two separate application runs. The two resource sets are bound into the DVC environment and assigned with the same set of virtual IP addresses (i.e., an identical namespace). This approach has several advantages including high application portability and runtime adaptation. Because the namespace is kept constant, the application can be independently run on the two resource configurations. Furthermore, to compete with asynchronous changes in resource availability and dynamic application requirements, the resources in the DVC environment can be transparently replaced. A virtual name whose presence remains intact can be associated with several distinct physical resources over time. From the application perspective, resources within the DVC environment are as reliable and easily accessible as in a local distributed environment.

In addition, the DVC model provides a set of group naming operations to simplify application management of collective communication and resources. At first, all resources are bound into a simple flat namespace. An application can create group, hierarchical, or other

naming structures for resources, providing easy group-based management. For example, the resources can be grouped based on their functions (e.g., storage, computing and visualization), communication roles (e.g., data source and sink), network locations, or other criteria. These groups can be grown, shrunk, subsetted or combined. This allows for flexible usage. Each group is associated with a unique logical name and IP address. The use of group names (or IP addresses) provides a natural basis for describing collective operations on resources. For instance, a multicast connection can be created between a virtual IP address of a resource and a group IP address.

4.3.3 Uniform resource access

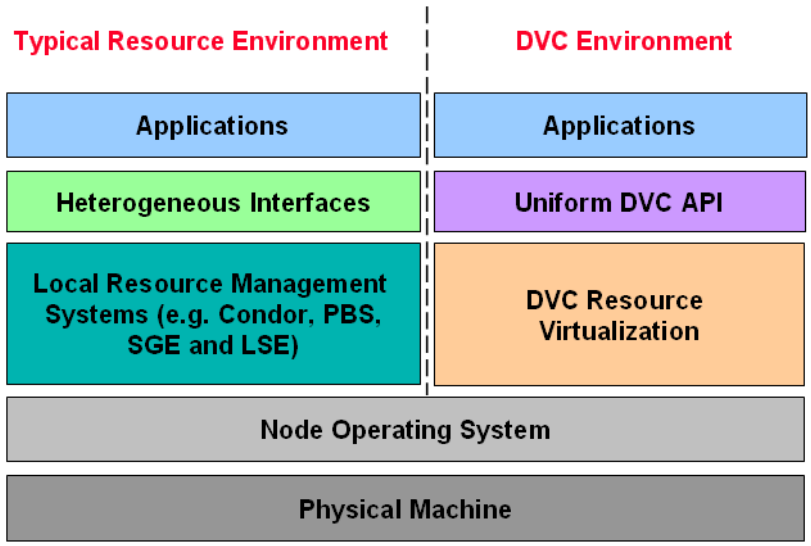


Figure 4-9: The DVC environment provides uniform access to distributed resources through virtualization

The DVC environment provides uniform access to Grid resources through virtualization, masking the complexity of site-specific management systems and resource heterogeneity. As shown in Figure 4-9, it provides a simple interface and virtualization services that application developers can use to directly access and manage remote resources.

Through this interface, applications only view simple virtual resources with the complexity of use comparable to that of a local cluster environment.

The DVC resource abstraction is realized by the cooperation of the virtualization services on individual resources. Lightweight control processes are created on them when new resources are allocated into the DVC environment. These processes enable a simple view of resources as they mask resource heterogeneity and complexity by providing a uniform way for the applications to interact. They in turn interact with physical resources via their native interfaces. Throughout application execution, these control processes also monitor resource utilization and availability, and may adjust the environment in response to dynamic application requirements and asynchronous changes in resource status.

4.3.4 Unified communication interfaces

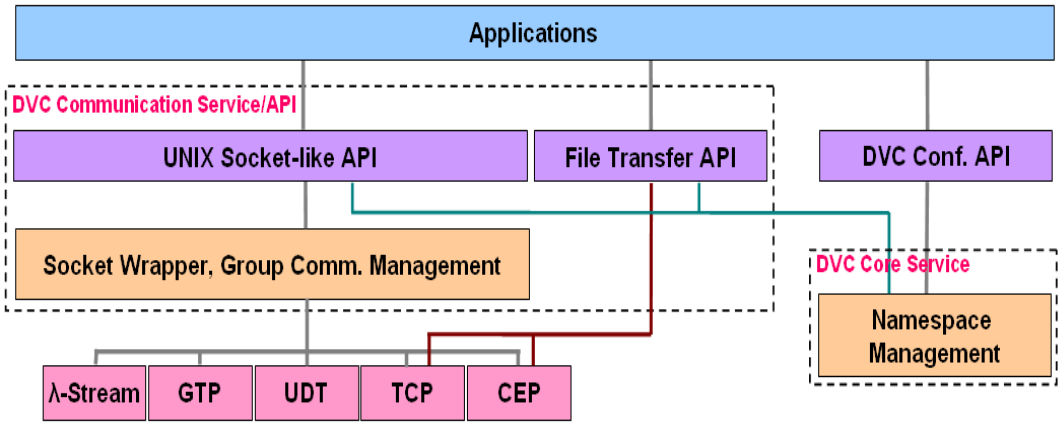


Figure 4-10: DVC high-speed communication architecture

The DVC environment provides a set of uniform communication interfaces to a range of novel transport protocols [23-25, 69, 70]. While these protocols are crucial for delivering the performance of high-speed, long distance connections, their diverse native interfaces complicate an application programming effort. To simplify application use of these protocols, we develop an integrated communication framework as shown in Figure 4-10. Here, the

implementations of individual protocols are provided as transport drivers. The framework integrates these drivers and presents uniform communication APIs to applications. These APIs are sock-like interfaces (*socket*, *listen*, *bind*, *connect*, *send*, *recv*, *sendfile*, *close*, etc.) which leverage the DVC virtual namespace and resource group names for simple expression of communication within the DVC environment. As virtual resource names (or IP addresses) are used for communication, the DVC implementation module translates them into physical IP addresses and calls appropriate protocol drivers. The use of these interfaces enables the applications constructed with the traditional socket interfaces to be easily ported to the DVC environment, thereby achieving the novel communication capabilities with minimum reprogramming effort.

4.4 Integrated Resource Selection

A significant problem in enabling high application performance and efficient resource use is the selection of appropriate sets of resources for individual applications. In Lambda-Grids, network configurability presents unique challenges for resource selection, adding the complexity of composed network connections to that of end resource selection. Here, we formulate the problem and present details of our combined resource selection approaches.

4.4.1 Problem Formulation

In the DVC model, applications describe and acquire combined sets of communication and end resources. As illustrated in Figure 4-11, the DVC-RCP is a resource planning service that takes a DVC-ISL specification and, subject to available network and grid resource information, matches it with an appropriate subset of the system resources. The DVC-ISL specification describes constraints on components (e.g., end resources, resource sets and communication nodes) and the DVC-RCP is responsible for matching these components

with the physical resources (e.g., computers, storage clusters and exchange switches). To realize application network connectivity requirements, the DVC-RCP selects a set of optical switches and links and composes them into satisfying network connections. This entire resource matching and network composition process is referred to as the *resource selection problem*.

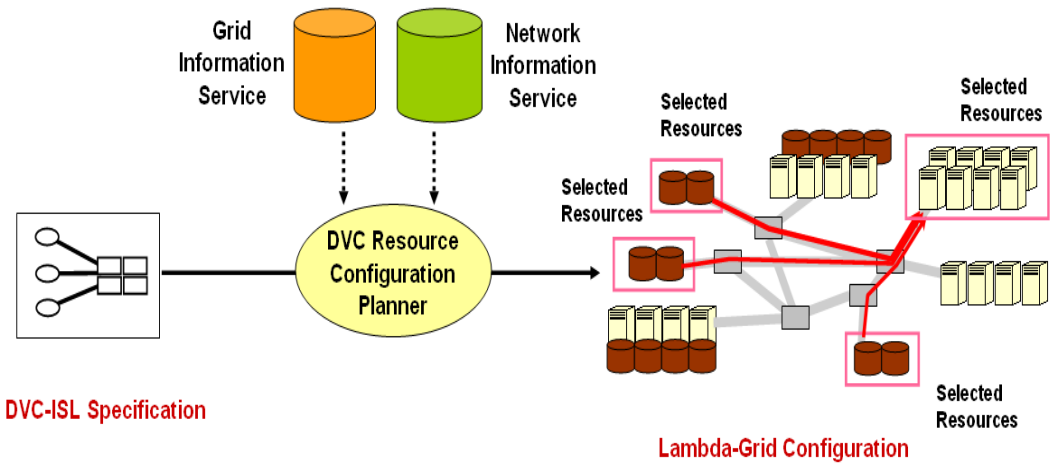


Figure 4-11: Selecting network and end resources to satisfy application needs

Our problem shares many similarities with resource selection problems for Grids [27, 53, 60] and network embedding problems [62, 63], proven as NP-hard. However, it differs from these in that it is not necessarily a one-to-one mapping between components in the application request and physical resources. For instance, required network connections are realized by sets of network switches and links. This kind of network planning is complex and is computationally harder than end-resource selection as it involves optimization of composed network resources.

Finding an acceptable solution that meets all application resource requirements is the primary goal. Two optimization objectives also need to be considered. The first is to select a

resource configuration that maximizes application capabilities. For many significant scientific applications, their performance is highly dependent on the sets of resources that host their computation and communication. Application performance may be dictated by a range of factors, depending on the application's characteristics. The DVC-ISL allows specification of a ranking function to guide choices of resource candidates that can make major application performance difference. Hence, the goal is to optimize the rank value of the solution. The other objective is to minimize network resource use in terms of total lambda (circuit) distance. This optimizes application communication latency while maximizing the probability of satisfying future application requests by conserving use of lambdas.

For large Lambda-Grids and/or complex application requirements, the resource selection task is very difficult. We consider large resource environments, comprised of millions of end resources (10^6 and more) and a few thousands of optical switches. Applications may also ask for a large number of indepdendent components which cannot be selected separately. Therefore, the problem is computationally hard and algorithms that guarantee optimal solutions are not practical at resource scale.

4.4.2 Current Practice: Separate Resource Selection

Traditionally, communication and end resources are managed separately by optical network and Grid middleware services. To acquire both types of resources, an application first queries a Grid resource broker [26, 27] for a set of available end resources and then contacts network services [17, 31, 32] to realize the desired network connections between them. If the chosen end resources are far apart, realizing the required connectivity may be impossible. To avoid repeated selection, the resource broker can produce multiple sets of end-resource candidates with good resource quality and the application presents these choices (as endpoints to be interconnected) to the network services. Finally, the network services return the resource

configuration (a combined set of end resources and private networks) with the lowest network cost. We refer to this approach as the *separate selection algorithm* (Sep). Figure 4-12 provides a short description of the algorithm.

Separate Resource Selection Algorithm (Sep)

1. For each component of a specification (an end resource or resource set), select *N* best candidates that satisfy all per-node and set attribute constraints that favorably contribute to a good rank value.
2. Among the chosen sets of candidates, enumerate all combinations to generate resource configurations and label them with their respective rank values
3. For each resource configuration, compose the required sets of network connections. If there is any unsatisfied connectivity constraint, reject that configuration
4. Among the accepted configurations, return the one with the best rank. Ties are broken in favor of the configuration with the lowest network cost.

Figure 4-12: Description of the separate resource selection algorithm

The above algorithm and close variants are employed in recent resource selection systems such as Virtual Grids [73], SWORD [60] and recipe-based service composition [61]. While these systems target resource environments with the traditional shared Internet, they separate the selection of end resources and the evaluation of network constraints in order to effectively solve the complex selection problems in a reasonable amount of time. This fast and simple approach should produce solutions with good resource quality enabling high application performance. However, it may incur high communication cost because end resources are chosen with little knowledge of available network connections. We use this separate selection approach as a baseline for our experiments.

4.4.3 Simulated Annealing Based Combined Resource Selection

While integrated resource selection provides opportunities for optimization of choices that span both communication and end resources, it is a computationally intensive task. An

exhaustive algorithm [61] that enumerates all possible combinations of resource candidates and guarantees optimal solutions are not practical at resource scale (thousands to millions of resources). Use of a randomized heuristic algorithm offers a good trade-off between selection optimality and cost; good algorithmic design will enable it to find near-optimal solutions in a modest amount of time. Our approach uses a randomized search heuristic based on simulated annealing (SA) with simultaneous end-resource selection and network configuration optimization. Figure 4-13 provides a high-level description of the *SA-based combined selection algorithm* (SA-Com).

SA-based Combined Resource Selection Algorithm

1. Initialize current temperature t_c, final temperature t_f, and number of iterations *numItr*
2. Generate an initial random resource configuration RC_0
3. Calculate the score of RC_0 using the "scoring function"
4. Repeat until $t_c < t_f$
    a. Use the "generation function" to generate a new resource configuration RC_{i+1} from RC_i
    b. Calculate the score of RC_{i+1} using the "scoring function"
    c. Generate a random number r on [0,1] and calculate the accept probability *accProb* using the "acceptance function"
    d. If RC_{i+1} has a "better score" than RC_i, or $r < accProb$, accept the transition from RC_i to RC_{i+1}. Otherwise, reject RC_{i+1}.
    e. If RC_{i+1} is the best solution thus far, let $RC_{best} \leftarrow RC_{i+1}$
    f. Update t_f using the "cooling schedule" after *numItr* iterations
5. If RC_{best} satisfies all constraints, return RC_{best}

Figure 4-13: Description of the SA-based combined resource selection algorithm

The SA algorithm is an iterative improvement search technique based on the physical process of "annealing" [74]. First, it produces a random resource configuration (RC_0) combining a set of chosen end resources and optical circuits. Then, the algorithm uses a *generation function* to control the transition from one configuration (RC_i) to another (RC_{i+1}), and uses a *scoring* function to determine the quality of a given configuration. If the new configuration (RC_{i+1}) has a better score than the previous one (RC_i), the transition is accepted.

Otherwise, it is accepted with some probability, as controlled by the current temperature (t_c). At first, the temperature is set to such a high value that nearly all new configurations are accepted. After a certain number of iterations, it is slowly reduced at the rate controlled by a *temperature cooling schedule*. When the current temperature is low enough, only better new configurations are accepted so the search is converted close to the optimal. Throughout the search process, the algorithm records the best-score configuration (RC_{best}) found thus far, and returns it when the current temperature drops below the pre-set final temperature (t_f) and if it meets all application resource requirements.

The design of the key functions of the algorithm, including the generation function, the scoring function, the acceptance function and the temperature cooling schedule are all significant to the quality of resource selection.

4.4.3.1 Generation Function

The function takes one resource configuration (RC_i) as input and modifies it to produce a new configuration (RC_{i+1}). Each resource configuration is sets of physical communication and end resources that are mapped to individual application components in the specification. In this way, a new configuration is generated by randomly picking one application component (c_k) and matching it with a new single or set of physical resource candidates, depending on the type of c_k. For example, “single-node”, “resource-set” and “cnode” components are matched with a storage server, compute cluster, and exchange switches, respectively. Further, if c_k has required network connectivity to any of other components, the function calls the network configuration planning module to compose necessary connections. To realize point-to-point connections, the module uses the Dijkstra shortest-path algorithm with the goal to minimize the distance of the composed optical circuit paths. Application communication latency as well as network resource efficiency is optimized

in this manner. To compose multicast connections, the module implements the "re-route-to-source" heuristic [75], shown to construct a multicast tree in constrained WDM networks with minimal average communication delay.

We maintain a list of application components and their respective domains – which represent sets of resource candidates that can be mapped to individual components. To derive the domain of each component, we use a node consistency algorithm [76] and rely on the database technique [77] to efficiently filter out those candidates that do not satisfy "per-node" constraints involving only that component. So that the search space is further reduced, we leverage the concept of "physical equivalence class" [62] to reduce the domain for each component. In a typical Grid environment, collections of homogenous resources are logically grouped into clusters. The resources within a cluster (such as computing nodes or storage systems), are generic and indistinguishable in terms of hardware and network properties. Because choosing any of these resources won't affect the quality of resource selection, we pick only one as representative and add it into the domain. Extra care must be taken when the same set of resources can be mapped to multiple application components. Here, we avoid adding the same resource candidates into the domains of different components.

4.4.3.2 Scoring Function

The function inputs a resource configuration and produces a score that represents its quality. The configuration with the best score is considered the best solution according to the three optimization objectives outlined in Section 4.4.1. These goals are: 1) to find a solution satisfying all application resource requirements; 2) to maximize application capabilities, as determined by a user-defined ranking function; and 3) to minimize network resource use (or conversely maximize network efficiency). Hence, we represent the score ($score_i$) as a three-tuple (sat_i, $rank_i$, $cost_i$), where sat_i is the total number of satisfied constraints, $rank_i$ is the rank

value and $cost_i$ is the network cost of the evaluated resource configuration (RC_i). We hold that $score_A = (sat_A, rank_A, cost_A)$ is better than $score_B = (sat_B, rank_B, cost_B)$, if any of the following three conditions are met:

(1) $sat_A > sat_B$, or
(2) $sat_A = sat_B$ and $rank_A > rank_B$, or
(3) $sat_A = sat_B$ and $rank_A = rank_B$ and $cost_A < cost_B$.

Inside the scoring function, there are two kinds of constraints evaluated. The first is a network connectivity constraint, including connectivity types and quality of service. The other is an end-resource attribute constraint involving more than one application components. Examples include the minimum total disk space, average CPU speed and compatible software of sets of machines. The function doesn't consider end-resource attribute constraints that involve only one component because they've already been evaluated during the discovery of the domain for each component. We characterize the network cost as the total weighted distance of all lambdas (optical circuits) of the evaluated resource configuration, or formally $\Sigma_i\ d_i*w_i$ where d_i is the distance and w_i is the number of allocated wavelengths of the optical circuit i.

The critical challenge in implementing this function is keeping the computational cost low. Because the function is called every time that a new resource configuration (RC_{i+1}) is generated, its computational cost contributes to a large fraction of the total running time of the selection algorithm. To minimize this cost, we take an advantage of the property that a new resource configuration (RC_{i+1}) is only slightly different from its base (or previous) configuration (RC_i). This is because the generation function picks one component and finds a new mapping for it. Instead of re-evaluating the number of satisfied constraints, rank value and network cost for all components, the function checks only those affected by the new mapping.

4.4.3.3 Acceptance Function

The function computes the probability of accepting a new resource configuration (RC_{i+1}) that has a worse score than its base one (RC_i). It is defined as *exp(-penalty/t_c)*, where t_c is the current temperature and *penalty* is the penalty value of the new configuration RC_{i+1}. The penalty is computed as *penalty = ((TotalConst – SatConst)/(TotalConst)*100)*, where *TotalConst* is the total number of constraints in the specification and *SatConst* is the number of satisfied constraints of RC_{i+1}.

4.4.3.4 Temperature Cooling Schedule

The cooling schedule controls how the temperature cools down and reaches the stop point of the algorithm. We use a simple geometric cooling function $t_c = t_c * \alpha$, where α is a cooling rate and t_c is the current temperature. The function is called every after *numItr* iterations, where *numItr* is the sum of the total number of candidates for each application component. The rationale is to allow consideration of every possible candidate for individual components before the temperature is reduced in each step. We set parameters α=0.95, starting temperature t_s=60, finish temperature t_f=0.75. These were chosen based on a large set of training experiments that allow the search to converge to a final RC close to optimal.

4.4.4 Top-down Hierarchical Combined Resource Selection

When we apply the SA-Com algorithm to large resource environments (comprised of millions of end resources and thousands of network switches), it can be expensive computationally. This is due to the immense solution space and a large number of circuit computations over large networks. To address this scalability issue, we develop a novel *top-down hierarchical combined selection algorithm* (Hier-Com) based on network clustering. As illustrated in Figure 4-14, the high-level concept is to obtain a sequence of successive

approximations of the original selection problem (via network clustering) until the size of the simplified problem is small enough to be directly solved by SA-Com. Then, the solutions to the simplified problems are used to prune candidates and drive resource selection in the more complex problems.

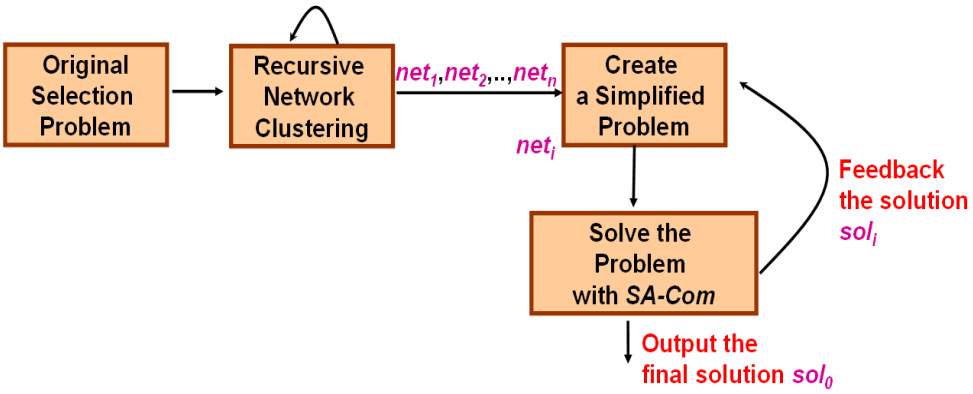


Figure 4-14: Functional flow of the top-down hierarchical combined selection algorithm

Top-down Hierarchical Combined Resource Selection Algorithm

1. Use the "network clustering function" to create abstract networks (net_1, net_2, …, net_n) at different hierarchical levels from an original networks (net_0)
2. Let $i \leftarrow n$
3. Repeat until $i = 0$
   g. Initialize a new simplified selection problem $prob_i$
      - Create a simplified resource environment from net_i
      - Use the "candidate filter function" that uses the solution sol_{i+1} to prune resource candidates
   h. Solve the problem $prob_i$ using the SA-Com algorithm and produce the solution sol_i
   i. $i \leftarrow i - 1$
4. Initialize a final simplified selection problem $prob_0$ from net_0
   j. Use the "candidate filter function" that uses the solution sol_1 to prune resource candidates
5. Solve the final problem $prob_0$ using the *SA-Com* algorithm and returns the final solution sol_0

Figure 4-15: Description of the top-down hierarchical combined selection algorithm

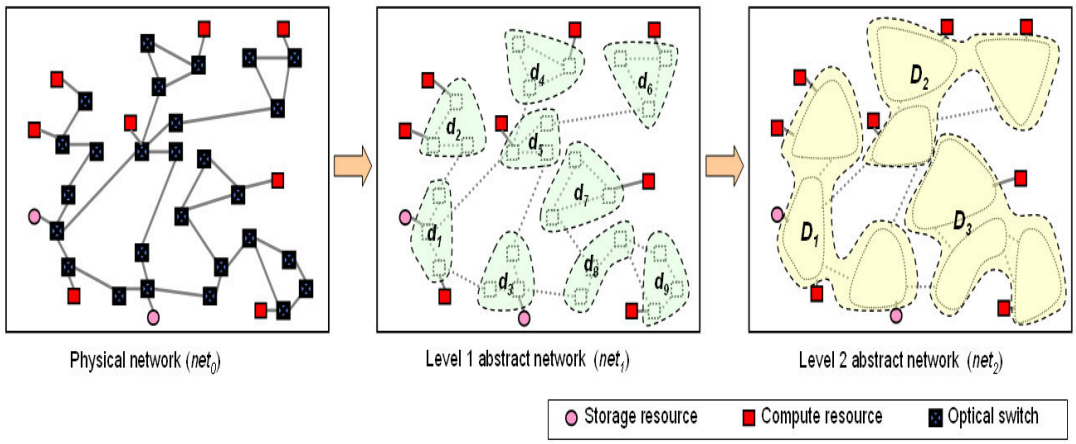


Figure 4-16: Creating abstract networks at different hierarchical levels with network clustering

Figure 4-15 provides a high-level description of the Hier-Com algorithm. It operates as follows. First, the *network clustering function* is used to create abstract networks (net_1, net_2, …, net_n) at different hierarchical levels. As shown in Figure 4-16, at each level we cluster the switches that are topologically close in the original physical network (net_0) into equal-size domains. The abstract network (net_i) consists of abstract switches representing these domains and abstract links summarizing the network connectivity between these domains. At higher hierarchical levels, the number of domains decreases and the domain size increases. Starting from the top level n, end resources are connected to the abstract switches that represent their domains – the resources are in the same domains as their physical switches. The resulting resource environment is a simplified selection problem ($prob_n$) – with a smaller size network (net_n) – that can be effectively solved by SA-Com. The solution (sol_n) indicates, for each application component, from which domain end resources will be chosen in the final result. Specifically, in sol_n, if the selected resource for a component c_k is in domain A that consists of switches 1-3, in the final result the selected resources for c_k will be chosen from only those candidates connected to switches 1-3. The rationale here is to approximate good selection of end resources by first using the coarse-grained, domain-level networks and later refining it.

Then, we proceed to use the abstract network in the next lower level (net_i) and exploit the previous solution (sol_{i+1}) to prune resource candidates from irrelevant domains with the *candidate filter* function. The result is a new simplified selection problem ($prob_i$) and is solved with SA-Com. This process is repeated until the lowest level is reached. As we proceed to more refined levels, domain size decreases and we can get more detailed information about the regions in the original network that the end resources will be picked from for each component. At the bottom level, we use the original network (net_0) and return the solution (sol_0) as a final result.

The effectiveness here depends on the design of the key functions of the algorithm, including the network clustering function and candidate filtering function. Following is a discussion of the details of these functions.

4.4.4.1 Network Clustering Function

The function accepts a parameter *clusterSize* and generates abstract networks at different hierarchical levels by recursively clustering switches into domains of *clusterSize* elements. As an example illustrated in Figure 4-16, we set *clusterSize* to 3, and it takes two iterations to cluster the original network (comprised of 27 optical switches) into two abstract networks (comprised of 9 and 3 domains). For the Hier-Com algorithm to be effective, switches that are topologically close together must be grouped into the same domain. This is because we use these domains to approximate communication cost between remote switches and assume good network connectivity within a domain.

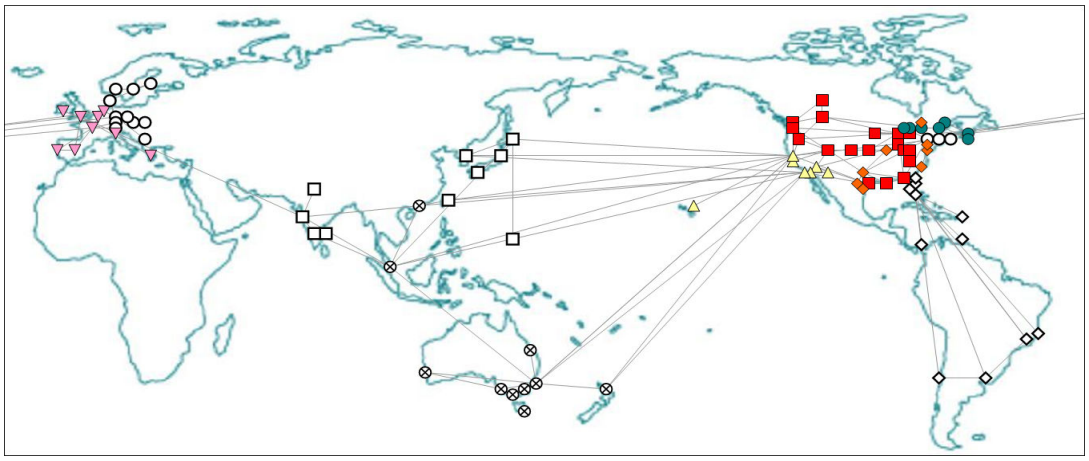
The network clustering can be modeled as a min-cut graph partitioning problem and solved with a generic graph partitioning algorithm. To solve it quickly and effectively, we exploit the METIS's state-of-the-art min-cut graph partitioning tool [78]. The challenge is how to construct an input graph to METIS by setting its edge weights appropriately so that the

resulting network is well clustered. In a graph-partitioning problem, the goal is the number of weighted edge-cuts to be minimized. In the network clustering problem, the objective is to place the switches with no or high-latency connections into different domains (or partitions). Therefore, we set edge weight to be inversely proportional to its corresponding link latency. Different approaches were explored and the following edge weight assignment formula was chosen:

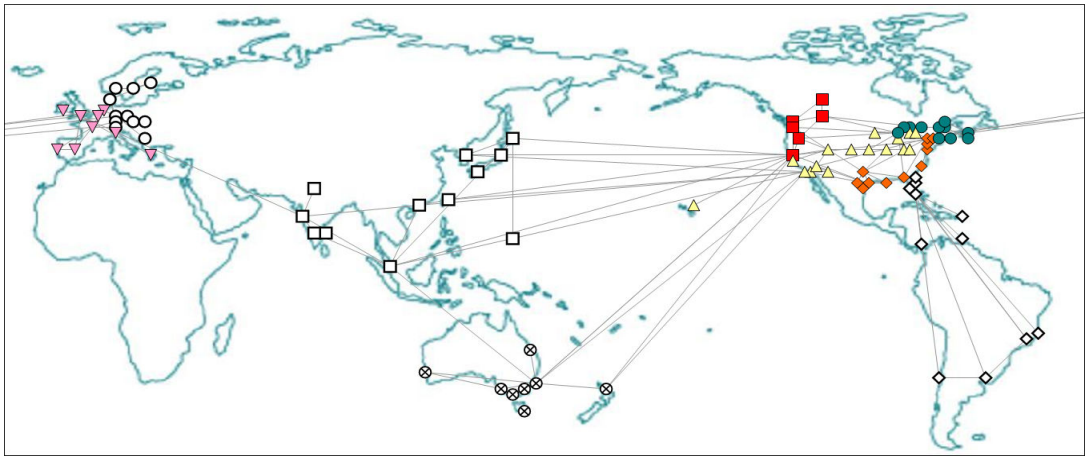
$$Weight\ (e_{ij}) = \left[\frac{(MaxLat - Lat\ (e_{ij}))}{(MaxLat)} \right] * 100 + 1$$

Here e_{ij} is a link (or edge) between two switches, *MaxLat* is the maximum latency of all links, $Lat(e_{ij})$ is the link latency and $Weight(e_{ij})$ is the edge weight of the link e_{ij}.

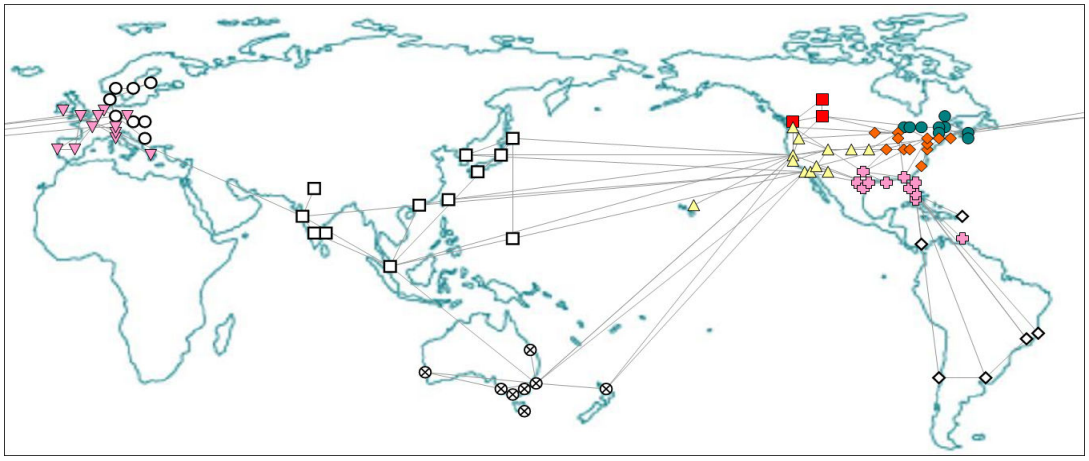
Figure 4-17 (a-c) give an exemplary result of partitioning the Verizon global network [79] with different edge weight assignment methods. In these figures, the different symbols represent the resulting domains assigned to switches after the network is partitioned. We see that our method (shown in Figure 4-17 (b)) produces the best result where the network is well partitioned based on geographical regions and the resulting domains have the least size variation. We also evaluate different edge weight assignment methods with other real ISP's metropolitan, national and global networks and find that our approach achieves better or comparable quality of the network partitioning results.



(a) $1/Lat(e_{ij})$



(b) $(((MaxLat - Lat(e_{ij}))/MaxLat) * 100) + 1$



(c) $(((MaxLat - Lat(e_{ij}))^2/(MaxLat)^2) * 100) + 1$

Figure 4-17: Partitioning the Verizon global network with different edge weight assignment methods

After the network is clustered (or partitioned) in each step, we create an abstract network where abstract switches represent the resulting domains and abstract links summarize the network connectivity between them. Our goal is to appropriately set the latency and bandwidth of each abstract link so it represents good approximation of the communication cost between remote switches and doesn't eliminate any feasible solution. During implementation, we approximate respectively the latency and available bandwidth of an abstract link by the average latency and sum bandwidth of all physical links across the two domains.

4.4.4.2 Candidate Filter Function

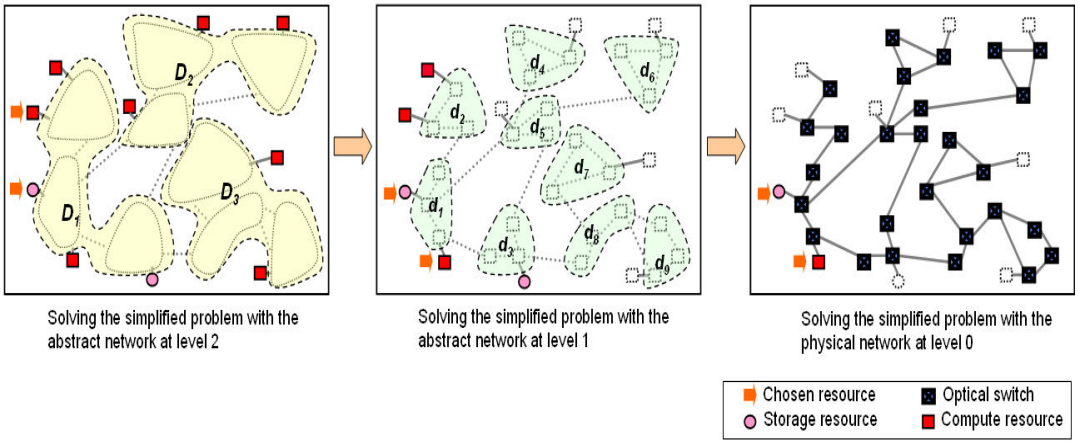


Figure 4-18: Pruning resource candidates and solving the simplified selection problem at different hierarchical levels

The function prunes resource candidates in the current abstract network (net_i) using the solution (sol_{i+1}) and information about the domains in the abstract network at the level above (net_{i+1}). The rationale is to make the complex selection problem solvable at resource scale (millions of end resources) by approximating good selection of end resources in a coarse-grained, domain-level network and using its solution to prune resource candidates from irrelevant domains. Specifically, for each application component c_k in the specification, we

filter out those candidates not in the same domain in the network net_{i+1} as the selected resources mapped to c_k in the solution sol_{i+1}.

As illustrated in Figure 4-18, we employ the Hier-Com algorithm to solve the problem of selecting one computer and one storage server with minimizing their communication delay the set goal. At first, the algorithm creates abstract networks at two hierarchical levels (see Figure 4-16). Starting from the top level, the algorithm calls the SA-Com subroutine to solve the simplified selection problem with the abstract network net_2. The solution sol_2 consists of one computer and one storage resource in the domain D_1. Then, the algorithm proceeds to the next level and uses sol_2 to filter out resource candidates from the other domains (D_2 and D_3). The result is the simplified resource selection problem with a much smaller solution space that can be effectively solved with SA-Com. This process is repeated until the bottom level is reached.

This resource candidate pruning technique has potential to significantly reduce the solution space and improve the running time of the Hier-Com algorithm. In a typical Grid environment, a large number of end resources (including computers and storage systems) are available across the wide area; though they are often generic and interchangeable. Hence, each application component usually has many candidates, ranging from hundreds to several thousands. Chapter 5 demonstrates the top-down hierarchical selection technique based on network clustering and candidate pruning greatly improve the selection time over SA-Com by 75.7% while maintaining good selection quality properties.

4.5 Prototype Implementation

Our key research contribution is the Distributed Virtual Computer (DVC), an integrated resource management framework enabling high application performance and resource efficiency in Lambda-Grids. Based on real use with scientific applications and

Lambda-Grid infrastructures, we develop a system software prototype implementing the DVC architecture. This section discusses several key prototype implementation issues.

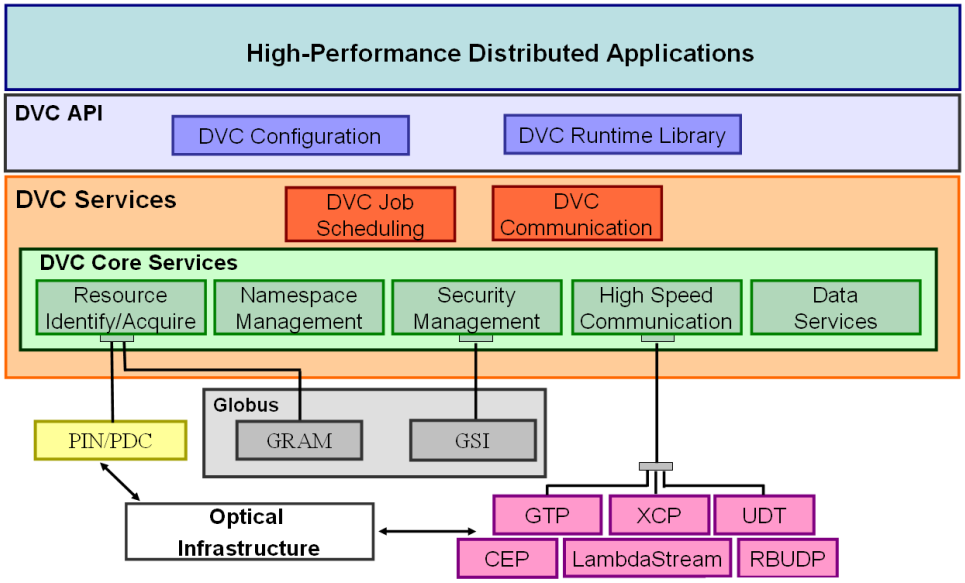


Figure 4-19: DVC system software architecture

As illustrated in Figure 4-19, the DVC framework is realized by a range of system software efforts in advanced distributed computing, high-speed communication, security management, and network control planes. Our approach is to leverage existing grid technologies for basic security and resource access, while being innovative to extract the novel capabilities of Lambda-Grids. In particular, we exploit the Globus's GRAM [37] to implement our resource binding service (DVC-RB) and rely on Grid Security Infrastructure (GSI) [34] for standard mechanisms for cross-domain authentication and authorization. We use the Photonic Interdomain Negotiator/Photonic Domain Controller (PIN/PDC) [31, 51] to dynamically configure a private network. Further, we implement a unified communication framework in Section 4.3.4 to integrate a range of high-speed transport protocols. These include TCP, GTP [23], CEP [25], UDT [24], etc. We leverage these grid and network service

components to implement DVC abstractions and make more application-oriented services available to developers.

The DVC prototype provides a set of simple interfaces (APIs) for application developers to create and manage a DVC environment. The package includes DVC configuration modules and runtime libraries with which applications can dynamically acquire/release resources, manage resource namespaces and security, submit jobs to execute on resources, as well as manage high-speed data transfers.

The following subsections describe in detail how the DVC:

- Manages and controls a DVC environment,
- Binds and manages end resources,
- Configures a private optical network, and
- Manages job execution.

4.5.1 DVC Environment Management

A DVC environment is a private computing environment simplifying the execution of distributed applications on Lambda-Grids. It provides a simple set of abstractions to enable complex collections of grid resources to be used in a fashion similar to that of a Storage Area Network (SAN) in terms of use and performance models. As shown in Figure 4-20, the DVC environment is realized by a group of daemon processes cooperatively running on a user's local host and remote end resources. When the user initiates a new DVC environment, a single daemon process, called a *DVC manager*, is created and associated with it. This manager is responsible for controlling and managing the DVC environment. It implements the resource configuration planner (DVC-RCP) and resource binder (DVC-RB) services that discover, acquire and bind resources into the DVC environment. The DVC manager also serves as domain security authority, managing trust relationships and implementing necessary security

policies and mechanisms. To foster robust applications, it monitors resources to detect asynchronous changes in resource performance and availability, and handles these events according to the previously agreed policies.

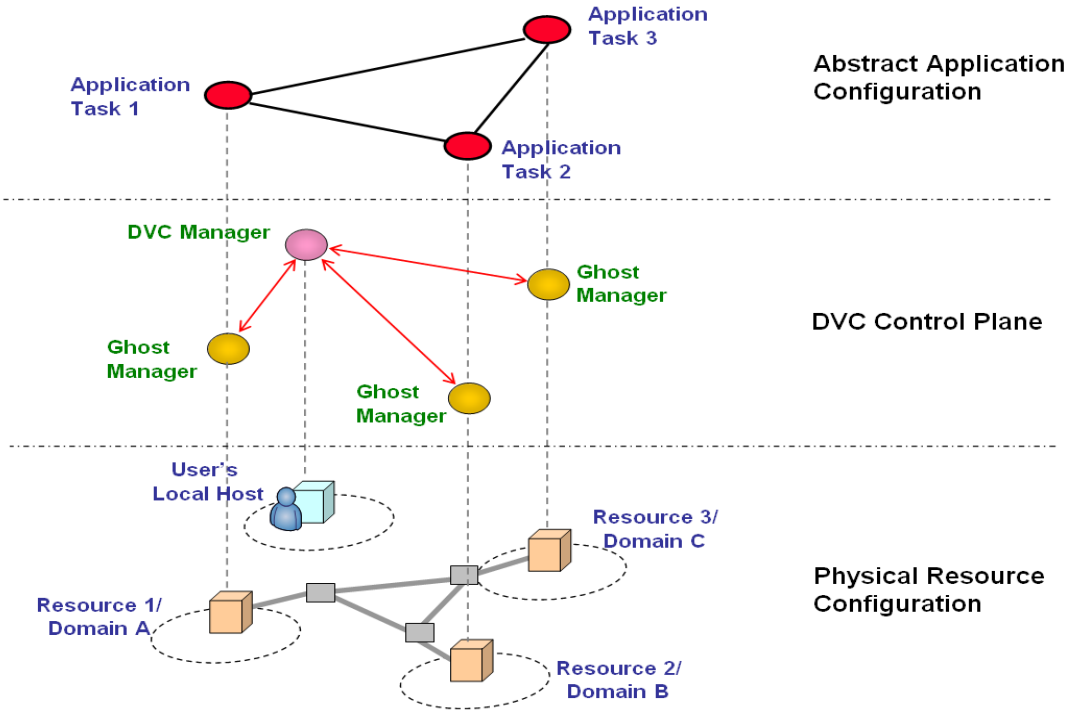


Figure 4-20: Implementing the DVC environment with a group of cooperative daemon processes (DVC manager and ghost managers)

When new end resources are bound into the DVC environment, the DVC manager initiates another lightweight daemon process, called a *ghost manager*, to run on each resource. The ghost manager has a major responsibility in visualizing a simple computing environment on the remote resource, serving as an intermediate layer between the application and the end resource. To simplify application management of naming, resource and communication, it implements a virtual namespace, unified communication interfaces, as well as mechanisms for uniform resource access. As well, the ghost manager periodically monitors and reports

resource status (including information about utilization, availability and active job processes) back to the DVC manager.

The DVC manager maintains a DVC descriptor that records the current DVC environment configuration and status. The descriptor includes information about DVC resources, the ghost managers on those resources, configured private networks, security and communication configuration, as well as active jobs. The DVC manager maintains a master copy of this descriptor and periodically updates its changes to the ghost managers.

4.5.2 End Resource Binding

The DVC system prototype can dynamically allocate and bind end resources into a DVC environment. As shown in Figure 4-21, to obtain access to remote resources, the DVC-RB service (implemented inside a DVC manager) authenticates to the site-specific resource managers under the user's identity, and initiates a DVC ghost manager on each resource. Subsequently, the ghost manager directly accepts commands from the DVC manager and performs necessary tasks (e.g., job invocation and job status inquiry) on the resource. In our implementation, we use Globus Resource Allocation Manager (GRAM) [37] as a secure gateway to the heterogeneous, site-specific resource managers, such as PBS [80] and SGE [81]. We use Grid Security Infrastructure (GSI) for user authentication and access authorization across administrative domains. Still, the DVC resource binding model is not limited to Globus GRAM/GSI. It can be easily applied to other resource management systems such as Condor [40].

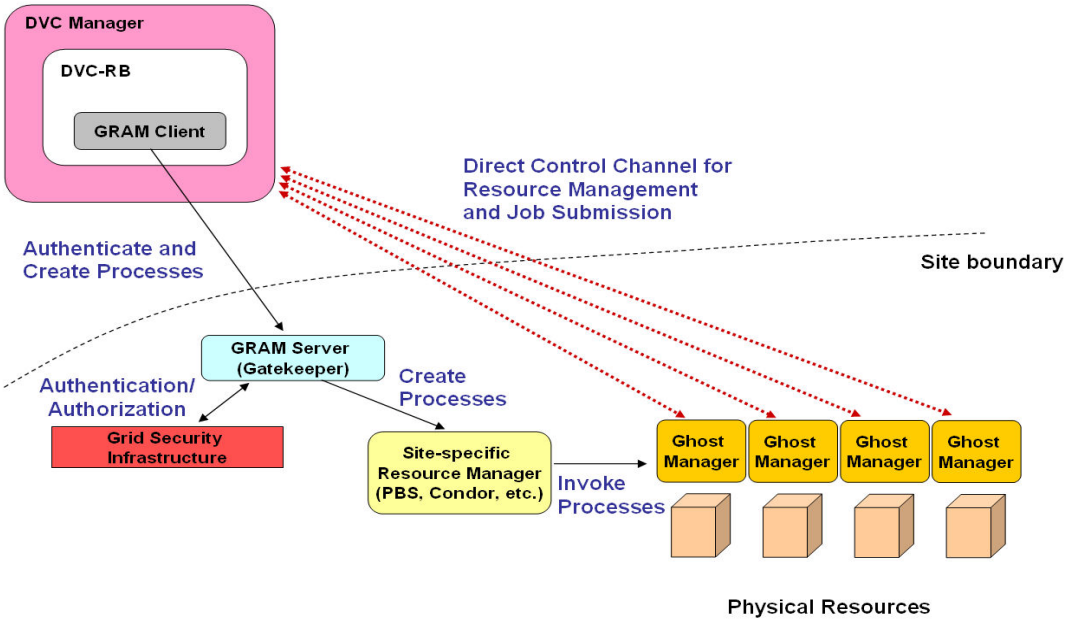


Figure 4-21: Binding remote resources into the DVC environment using Globus GRAM/GSI

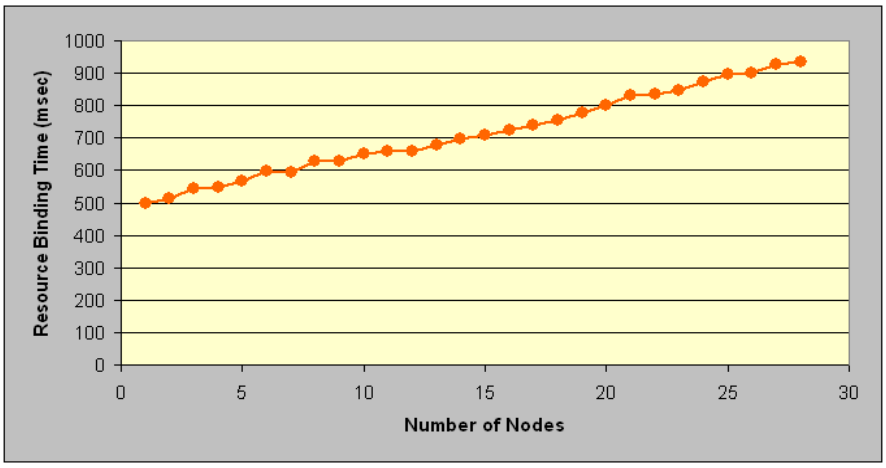


Figure 4-22: Resource binding overhead as a function of the number of resources

A key challenge in implementing the DVC-RB is rapidly binding a large number of end resources into a DVC environment. Many applications require tens to hundreds of resources for large-scale computation; they must allocate them quickly so as to minimize the turnaround time. Our approach is to bind multiple resources in one collective operation and

employ non-blocking calls to GRAM servers. This allows for multiple outstanding requests to remote GRAM servers and reduces the total resource binding time. Figure 4-22 shows the resource binding overhead as a function of the number of resources to bind in a local cluster. As can be seen, 28 resources were bound in less than a second. This overhead is typically very small, given that most large-scale scientific and engineering applications run [2,3] for several hours or longer.

4.5.3 Optical Network Configuration

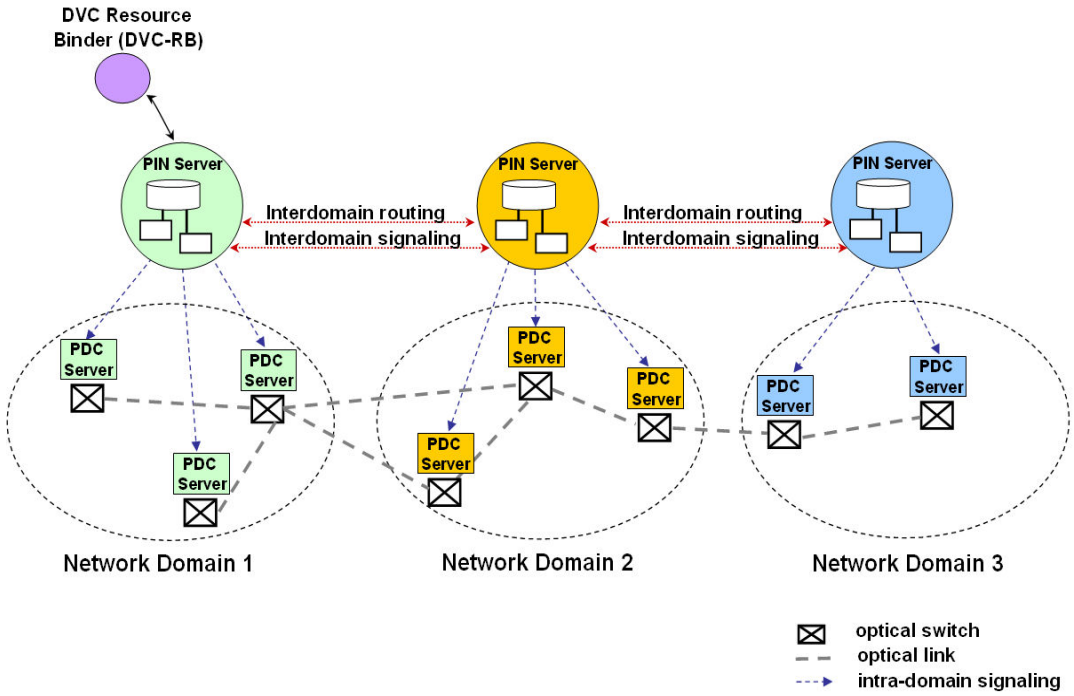


Figure 4-23: The DVC system software implementation exploits the PIN/PDC for configuring a private optical network across domains

The DVC system prototype has the ability to configure optical circuit paths (lightpaths) dynamically to realize private networks for high bandwidth, low jitter communication. These paths may interconnect geographically distributed end resources across multiple networks and administrative domains. To obtain a desired network configuration, the DVC resource binder (DVC-RB) employs the Photonic Interdomain Negotiator (PIN) [31], a

distributed control architecture for dynamic lightpath provisioning. The PIN addresses the complex issues of interdomain routing, signaling and security for lightpath scheduling and configuration. As shown in Figure 4-23, the PIN architecture is composed of distributed PIN servers across heterogeneous networks. One server is required and responsible for each network domain. For each optical path to configure, the DVC-RB passes the pair of endpoint references (e.g., public IP addresses of end resources) together with the required connectivity properties (e.g., bandwidth) to the local PIN server. In response, the server propagates the request message to remote domains until the destination is reached. For each domain along the path, the corresponding PIN server translates the request into an intradomain signaling message and dispatches it to a set of Photonic Domain Controller (PDC) servers [51] which control individual optical switches. Finally, each PDC server modifies the configuration of its local switch to establish the requested optical path. In our implementation, the network path configuration operation is atomic. If any subset of optical paths cannot be created, all pre-established paths are released and the failure notice is reported back to the application.

So that it provides guaranteed high-speed communication performance, the DVC model dedicates a set of optical circuit paths to each application throughout its execution time. The allocated optical paths are reserved for private use until the DVC environment terminates (unless the circuit path release is explicitly requested by the application). To release the reserved paths, the DVC-RB makes a request to the local PIN server. The request is then forwarded through different network domains and the responsible PDC servers ultimately release the configured circuits.

Although the current DVC system prototype utilizes the PIN/PDC for dynamic network configuration, it is not limited to it. The DVC model is also compatible with a wide range of other dynamic lightpath provisioning tools such as UCLP [32], DRAGON [17], DRAC [82] and Bigbangwidth [83].

4.5.4 Job Execution Management

The DVC prototype supports reliable execution of an application's jobs. It schedules jobs to run on distributed end resources under the DVC environment based on their priority, submission time, or other applicable policies. It also mediates communication among remote jobs and supports job restarts and migration in case of unexpected execution failure or resource unavailability.

Table 4-1: A list of environment variables that can be specified in a job specification

Variable Name	Value Type	Description
Remotedir	string	A working directory location at remote resources
Executable	string	A path to an executable file at remote resources
Numproc	integer	The number of processes to invoke for this job
Stdin	string	A standard input of the job process(s) on remote resource
Stdout	string	A standard output of the job process(s) on remote resources
Stderr	string	A standard error of the job process(s) on remote resources
Arguments	list of string(s)	Input arguments of the job process(s) on remote resources
Environment	list of string(s)	Unix environment variables to set before launching job process(s)
Ifiles	list of string(s)	A list of files to transfer from a local host to remote resources before launching job process(s)
Ofiles	list of string(s)	A list of files to transfer from remote resources to a local host after job processes(s) is completed

To submit a job to run, the user creates a job specification describing the characteristics and desired execution environments. Table 4-1 lists the environment variables that can be specified. When an application is started, the DVC manager selects computing resources from the DVC resource pool and binds them up with the application. When the application spawns computing tasks to run on remote resources, the DVC manager directly contacts the corresponding ghost managers (bypassing GRAM and other site-specific resource managers) to invoke processes on the chosen resources.

4.5.5 Integrated Communication Library

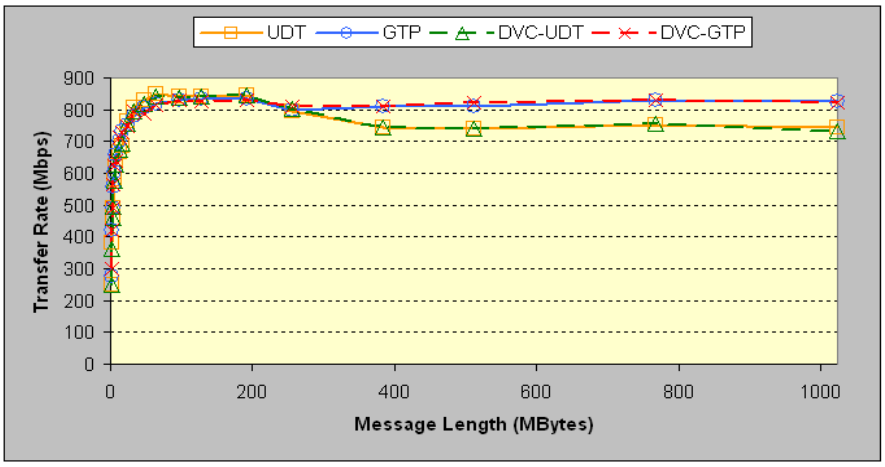


Figure 4-24: Comparison of the transfer rate between different transport protocols with and without the wrapper module with varying message size

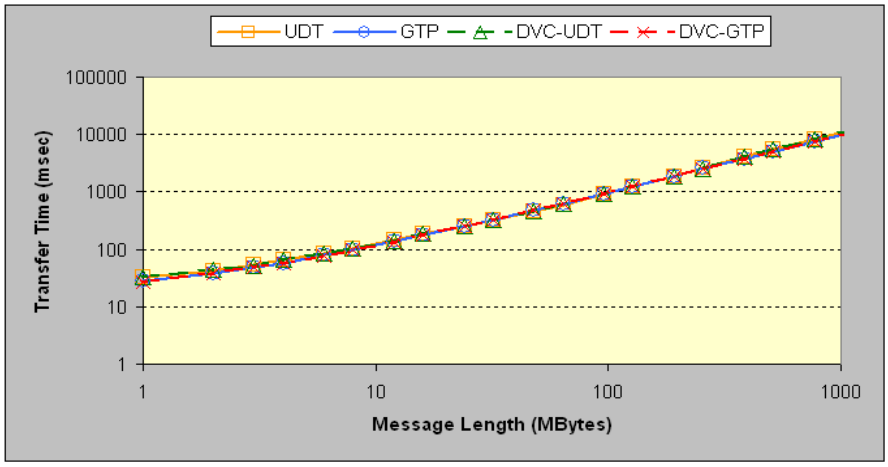


Figure 4-25: Comparison of the transfer time between different transport protocols with and without the wrapper module with varying message size

We implement the DVC communication architecture (see Figure 4-10) that provides a set of uniform interfaces to the novel transport protocols (including GTP, UDT and CEP). We import these protocols in the form of transport drivers and implement a lightweight interface wrapper module. Extra care must be taken in the implementation that the wrapper module must not incur a significant overhead to the original protocol performance, including

achievable transfer rate and communication latency. A simple experiment verifies this where we set up two dedicated hosts on a local cluster and ran memory-to-memory data transfers with different transport protocols with and without the wrapper module. We varied the message length, measured the transfer time, and calculated the transfer rate.

Figure 4-24 and 4-25 compares the resulting transfer rate and sending time when we use GTP and UDT with and without the DVC wrapper module. Our results show the wrapper module doesn't incur any noticeable overhead for all message size. For each protocol, we achieve the comparable transfer performance for the runs both with and without the wrapper module.

4.6 Summary

In this chapter, we presented several DVC system design and implementation issues. We first introduced the DVC architecture (service model) and its key components. We presented the DVC Integrated Specification (DVC-ISL) that allows explicit description of application communication requirements. We discussed a set of simplifying DVC abstractions enabling the complex Lambda-Grid environments to be used in a fashion comparable to a private resource workgroup. We then presented two novel combined resource selection algorithms (SA-Com and Hier-Com) to optimize application capabilities and network resource efficiency in Lambda-Grids. Lastly, we discussed the key implementation issues of the DVC system prototype.

4.7 Acknowledgement

Chapter 4, in part, is published as “Distributed Virtual Computer (DVC): Simplifying the Development of High Performance Grid Applications” by Nut Taesombut and Andrew A. Chien in the proceedings of the Workshop on Grids and Advanced Networks (GAN’04), April 2004. The dissertation author was the primary researcher and co-author of this paper.

Chapter 5. Evaluating Resource Selection Strategies

We have presented the DVC that provides a simple service model for applications and allows for coordination between communication and end resources to achieve guaranteed application performance. A key element of the DVC system is the DVC-RCP (resource configuration planner) that combines the selection of network and end resources to enable high application capabilities and network resource efficiency. As proof, we evaluate different resource selection strategies via simulation across a wide range of realistic application models and resource environments. Our metrics include success ratio, selection cost, resource quality, application lambda distance, system throughput and network utilization.

This chapter is organized as follows. Section 5.1 describes our evaluation methodology, including a simulation model, resource environments, application models and evaluation metrics. The quality of results that each selection algorithm can achieve and the selection cost is evaluated in Section 5.2. Section 5.3 looks at how different selection algorithms affect resource utilization and system throughput. Finally, we summarize our simulation results in Section 5.4.

5.1 Methodology

We compare and evaluate three resource selection algorithms (separate selection (Sep), SA-based combined selection (SA-Com) and hierarchical combined selection (Hier-Com) via simulations, synthetic application request workloads and realistic Lambda-Grid resource configurations. The following key research questions are of particular interest in this evaluation:

- What solution quality can each selection algorithm achieve (in terms of end-resource quality and network resource efficiency)?

- How often does each algorithm succeed or fail (as a function of application request complexity and resource configuration)?
- How fast does each algorithm run? How does the runtime scale with application request complexity and Lambda-Grid size?
- How do resource selection decisions of each algorithm shape overall system resource utilization and throughput?
- How do the combined selection algorithms and the separate selection algorithms compare overall?

In this empirical study, the following components are used to construct our experiments:

- **Synthetic workloads** modeled after realistic applications targeting Lambda-Grids, including high-performance computing, collaborative and remote data visualization, and distributed content delivery applications.
- **Synthetic Lambda-Grid configurations** based on state-of-the-art research tools for generating communication and Grid resources (cluster and host information) with the distribution matching the currently deployed ISP network topologies and computational Grids.

For all experiments, we ran simulation on a single compute machine. Table 5-1 shows its detailed specification. We employed the MySQL database [77] to store and provide Grid and network resource information. For each experiment, we ran resource selection and the MySQL server as two separate threads.

Table 5-1: Detailed specifications of the compute machine for simulation study

Attribute Name	Attribute Value
Number of processors	4
Processor model	Intel Xeon 2.8 GHz
Hardware platform	i686
Cache size	512 KB
Physical memory	2 GB
Disk space	210 GB
Operating system	GNU/Linux
Kernel release	2.6.9-22.ELsmp

5.1.1 Lambda-Grid Configurations

In order to assess selection algorithms in resource environments that scale past the complexity of the currently deployed Lambda-Grids, we use synthetic resource configurations. The simulated infrastructures are comprised of optical circuit-switched networks and Grid resources.

Table 5-2: Details of the studied multi-domain, global network topology

Internet Service Providers (ISPs)	Network Presence	Number of PoPs	Number of Links	Average Edge Degree	Average Link Latency (msec)
AT&T	Global	115	170	2.957	6.814
British Telecom	Global	109	189	3.468	6.625
Cogent	North America/ Europe	87	100	2.299	1.921
Global Crossing	Global	329	389	2.347	1.552
Level3	USA/Europe	59	97	3.288	2.283
NTT/Verio	Global	39	74	3.795	13.324
Qwest	USA	53	99	3.736	2.493
Sprint	Global	282	379	2.688	3.214
Time Warner	USA	48	73	3.042	2.110
Verizon	Global	98	187	3.816	7.950
Interdomain Links			597		5.400
Total		1219	2351	3.857	4.590

We consider two types of realistic optical network topologies. First, we use a real map of the multi-carrier Internet backbone network, consisting of ten leading Internet Service Providers (ISPs) in the world – AT&T [84], BT [85], Cogent [86], Global Crossing [87], Level3 [88], NTT/Verio [89], Qwest [90], Sprint [91], Time Warner [92] and Verizon [79].

We derived the current network topology of individuals ISPs from their company websites and then inferred their peering points from the Rocketfuel's traceroute data [93, 94] (generated by 300 traceroute servers across the globe). To reduce the number of traces that needed to be looked up, we used the AS-peering relationships information published by CAIDA [95, 96], and selected only those traceroutes that traverse pairs of the peering ISPs. Such map is a PoP-level network topology where each Point-of-Presence (PoP) represents a network access point in a city, and we substituted these access points with optical switches. To derive the latency of a link between two PoPs, we determined the latitude and longitude of their geographical presence, calculated their distance using the great circle method [97], and computed the latency using this distance divided by the speed of light. The resulting topology has 1,219 switches and 2,351 links; the details are given in Table 5-2. While we consider only a small number of ISPs, they are the dominant network service providers in the world and account for a large fraction of today's optical fiber infrastructures.

Second, in order to evaluate the scalability of different algorithms, we used the BRITE topology generator [98] to generate large multi-domain network topologies with varying numbers of switches (from 1,000 to 10,000). We carefully chose BRITE's parameters so that the edge degree and average link latency of the generated topologies are close to those of the real Internet backbone network above. For all networks, we assigned 20 lambdas, each at 1 Gbps, to individual links.

Our Grid (end) resource model is based on recent work [33] on realistic resource modeling for today's computational Grids, such as GriPhyN [13], TeraGrid [11], iVDGL [12] and EU-DataGrid [14]. For each derived network above, we generated synthetic end resources (cluster and host information) and randomly assigned them to individual switches. Each switch is connected to 4 clusters (or 268 end resources) on average. Each end resource was given a unique IP address and is connected to its switch via a 10 Gb/s uplink.

Table 5-3: Details of the studied Lambda-Grid configurations

Configuration ID	Number of ISPs	Number of Switches	Number of Links	Average Link Latency (msec)	Number of End Resources
Internet Backbone	10	1,219	2,351	3.857	331,203
Brite-1000	10	1,000	2,030	3.675	272,476
Brite-2000	20	2,000	4,080	3.735	532,771
Brite-4000	40	4,000	8,160	3.699	1,069,288
Brite-6000	60	6,000	12,240	3.671	1,596,973
Brite-8000	80	8,000	16,320	3.728	2,126,462
Brite-10000	100	10,000	20,400	3.713	2,769,338

Table 5-3 summarizes the resulting Lambda-Grid configurations. Our largest resource environment is comprised of 10,000 optical switches and 2.8 millions of end-resources, which is significantly larger than existing Lambda-Grid infrastructures, such as OptIPuter [15], CHEETAH [18] and DRAGON [17].

5.1.2 Application Models

In our evaluation we use synthetic workloads derived from a range of realistic application models targeted for Lambda-Grids. These includes: 1) high-performance distributed computing; 2) collaborative and remote data visualization; and 3) distributed content discovery. The details of the request workloads for these applications are discussed below.

5.1.2.1 High-Performance Distributed Computing

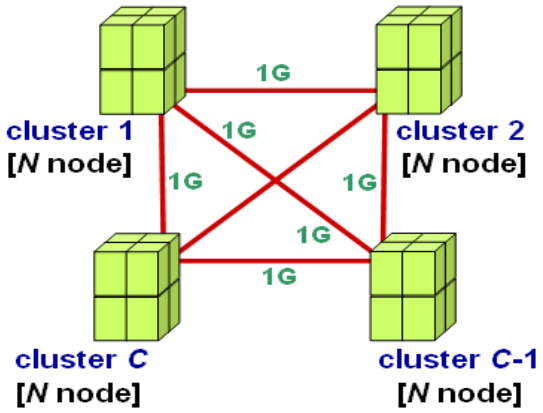


Figure 5-1: A resource request for high-performance distributed computing applications; (*ClusterSet*(4,8))

```
ClusterSet_4_8 =[
  cluster1 ISA SET [CPUSpeed > 2.5; MemoryFree > 3296; DiskFree > 150]; Count(cluster1)==8;
  cluster2 ISA SET [CPUSpeed > 2.5; MemoryFree > 3296; DiskFree > 150]; Count(cluster2)==8;
  cluster3 ISA SET [CPUSpeed > 2.5; MemoryFree > 3296; DiskFree > 150]; Count(cluster3)==8;
  cluster4 ISA SET [CPUSpeed > 2.5; MemoryFree > 3296; DiskFree > 150]; Count(cluster4)==8;
  conn1 ISA CONN (<cluster1>) [type = "intra-cluster"; Bandwidth >= 1000];
  conn2 ISA CONN (<cluster2>) [type = "intra-cluster"; Bandwidth >= 1000];
  conn3 ISA CONN (<cluster3>) [type = "intra-cluster"; Bandwidth >= 1000];
  conn4 ISA CONN (<cluster4>) [type = "intra-cluster"; Bandwidth >= 1000];
  lambda1 ISA CONN (<cluster1>, <cluster2>) [type = "lambda"; Bandwidth >= 1000; Latency < 60];
  lambda2 ISA CONN (<cluster1>, <cluster3>) [type = "lambda"; Bandwidth >= 1000; Latency < 60];
  lambda3 ISA CONN (<cluster1>, <cluster4>) [type = "lambda"; Bandwidth >= 1000; Latency < 60];
  lambda4 ISA CONN (<cluster2>, <cluster3>) [type = "lambda"; Bandwidth >= 1000; Latency < 60];
  lambda5 ISA CONN (<cluster2>, <cluster4>) [type = "lambda"; Bandwidth >= 1000; Latency < 60];
  lambda6 ISA CONN (<cluster3>, <cluster4>) [type = "lambda"; Bandwidth >= 1000; Latency < 60];
  Maximize((Avg(cluster1.CPUSpeed) + Avg(cluster2.CPUSpeed) + Avg(cluster3.CPUSpeed) +  Avg(cluster4.CPUSpeed))/4)
]
```

Figure 5-2: A DVC-ISL resource specification for high-performance distributed computing applications; (*ClusterSet*(4,8))

A popular and significant application model for Lambda-Grids is high-performance distributed computing (HPDC). HPDC is employed extensively by many scientific and engineering applications, such as high-energy physics [7], seismic processing [99], financial analysis [100], climate modeling [101] and bioinformatics/genetic research [102]. These

examples all involve large-scale computations and massive data transfers. Typically, HPDC applications require a large set of compute resources, as well as high-throughput and low-latency communication services [103]. Therefore, we model the resource request for these applications as *ClusterSet*(*C*, *N*) or tightly-coupled sets of compute clusters shown in Figure 5-1. A sample DVC-ISL specification for *ClusterSet*(4,8) is illustrated in Figure 5-2. Specifically, it requests for *C* compute clusters, each with *N* nodes, and lambda connectivity between each pair of clusters. Each compute node has three desired attributes: CPU speed, physical memory, and disk space. In order to investigate their impact on the quality of resource selection, we varied the required values of these attributes as shown in Table 5-4.

Table 5-4: Required resource attributes of compute clusters for high-performance distributed computing applications

Attribute Name	Distribution	Low End	High End	Distribution biased toward
Minimum CPUSpeed	Zipf	1.0 GHz	3.0 GHz	high end
Mimimum MemoryFree	Zipf	512 MB	4096 MB	high end
Minimum DiskFree	Zipf	80 GB	160 GB	high end
Minimum Intracluster Communication Speed	Constant	1 Gbps	1 Gbps	-

The required lambda connectivity is 1 Gb/s of bandwidth and less than 60 msec of latency. For many HPDC applications, their performance is highly influenced by the computing speed of the resources hosting their computations. Therefore, we model the ranking function of these applications as to maximize the average CPU speed of the allocated compute clusters. To scale request complexity, we increase the number of required clusters (*C*) as well as the number of nodes per cluster (*N*).

5.1.2.2 Collaborative and Remote Data Visualization

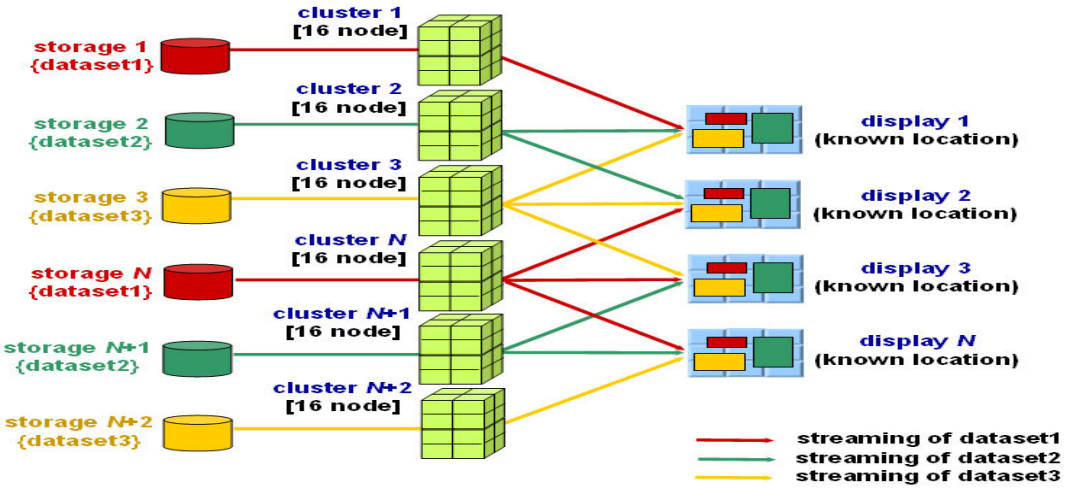


Figure 5-3: A resource request for collaborative and remote data visualization applications; (*DataViz*(*N*))

```
DataViz_2 =[
   display1 ISA [Hostname == “10.4.84.78”]; display2 ISA [Hostname == “10.29.168.82”];
   storage1 ISA [InSet(DataSet, "data_2343 ")];  storage2 ISA [InSet(DataSet, "data_78 ")];
   storage3 ISA [InSet(DataSet, "data_108")];    storage4 ISA [InSet(DataSet, "data_2343 ")];
   cluster1 ISA SET [CPUSpeed > 2.4; MemoryFree > 4096; DiskFree > 160]; Count(cluster1)==16;
   cluster2 ISA SET [CPUSpeed > 2.4; MemoryFree > 4096; DiskFree > 160]; Count(cluster2)==16;
   cluster3 ISA SET [CPUSpeed > 2.4; MemoryFree > 4096; DiskFree > 160]; Count(cluster3)==16;
   cluster4 ISA SET [CPUSpeed > 2.4; MemoryFree > 4096; DiskFree > 160]; Count(cluster4)==16;
   conn1 ISA CONN (<cluster1>) [type = "intra-cluster"; Bandwidth >= 1000];
   conn2 ISA CONN (<cluster2>) [type = "intra-cluster"; Bandwidth >= 1000];
   conn3 ISA CONN (<cluster3>) [type = "intra-cluster"; Bandwidth >= 1000];
   conn4 ISA CONN (<cluster4>) [type = "intra-cluster"; Bandwidth >= 1000];
   lambda1  ISA CONN (<cluster1>, <display1>) [type = "lambda"; Bandwidth >= 4000];
   lambda2  ISA CONN (<cluster2>, <display1>) [type = "lambda"; Bandwidth >= 4000];
   lambda3  ISA CONN (<cluster3>, <display1>) [type = "lambda"; Bandwidth >= 4000];
   lambda4  ISA CONN (<cluster2>, <display2>) [type = "lambda"; Bandwidth >= 4000];
   lambda5  ISA CONN (<cluster3>, <display2>) [type = "lambda"; Bandwidth >= 4000];
   lambda6  ISA CONN (<cluster4>, <display2>) [type = "lambda"; Bandwidth >= 4000];
   lambda7  ISA CONN (<storage1>, <cluster1>) [type = "lambda"; Bandwidth >= 2000];
   lambda8  ISA CONN (<storage2>, <cluster2>) [type = "lambda"; Bandwidth >= 2000];
   lambda9  ISA CONN (<storage3>, <cluster3>) [type = "lambda"; Bandwidth >= 2000];
   lambda10 ISA CONN (<storage4>, <cluster4>) [type = "lambda"; Bandwidth >= 2000];
   Maximize((Avg(cluster1.CPUSpeed) + Avg(cluster2.CPUSpeed) + Avg(cluster3.CPUSpeed) +  Avg(cluster4.CPUSpeed))/4)
]
```

Figure 5-4: A DVC-ISL resource specification for collaborative and remote data visualization applications; (*DataViz*(2))

Collaborative and remote data visualization is becoming increasingly important for many scientific fields, including oceanography [6], biomedical [1] and earth science [4]. This application enables researchers from distant institutions to interactively visualize and analyze very large data objects in real-time [3]. This enhances the understanding of complex scientific systems. We model the resource request *DataViz(N)* for collaborative data visualization as shown in Figure 5-3. Figure 5-4 shows a sample DVC-ISL specification for *DataViz(2)*. Specifically, scientists at *N* known remote locations (on the right of the figure) want to simultaneously visualize three datasets (shown in red, yellow and green boxes) on their local displays. The visualization of each dataset is driven by one rendering cluster of 16 nodes. The required resource attributes of the rendering clusters are given in Table 5-5.

Table 5-5: Required resource attributes of rendering clusters for collaborative and remote data visualization applications

Attribute Name	Minimum Requirement
CPUSpeed	2.4 GHz
MemoryFree	4096 MB
DiskFree	160 GB
Minimum Intracluster Communication Speed	1 Gbps

Each cluster is capable of handling only one dataset, but can support up to three remote displays. As illustrated in Figure 5-3, to support *N* displays, it requires *N*+2 rendering clusters. The required network connectivity between each pair of the displays and rendering clusters is 4 Gb/s of bandwidth and less than 60 msec of latency. The request also requires each cluster to have 2 Gb/s lambda connectivity to a storage server which stores a replica of a certain dataset to visualize. The ranking function for this application is to maximize the average CPU speed of the rendering clusters. To scale request complexity, we increase the number of display locations (*N*).

5.1.2.3 Distributed Content Delivery

```
DeliveryRequest =[
  client ISA [Hostname == “132.239.226.35”];
  storage ISA [InSet(DataSet, "data_410”)];
  lambda ISA CONN (<client>, <storage>) [type = "lambda"; Bandwidth >= 1000]
  Minimize(lambda.Latency)
]
```

Figure 5-5: A DVC-ISL resource specification for a content delivery request

A distributed content delivery application was chosen because it shares many aspects with “scientific data distribution and sharing” [1], which features large collections of distributed data objects and on-demand communication, and is a dominant large-scale scientific application targeting Lambda-Grids today. Our workload for this application are synthetic traces of content delivery requests where each request is a client at a known location requesting for the streaming of a certain movie object via a 1 Gb/s private network. Initially, a collection of movie objects were replicated and distributed across a set of replica servers. The ranking function for this application is to minimize the communication latency between the client and the chosen replica server.

5.1.3 Replicating and Distributing Data Objects

To support the application models above, we assume each Lambda-Grid configuration contains a set of replica servers. Each server maintains a collection of (movie) data objects. The replica servers with certain data objects are key requirements of collaborative data visualization and distributed content delivery applications (See Figure 5-4 and Figure 5-5). Our approach uses the following realistic model based on recent research [104, 105] to replicate and distribute data objects among the replica servers.

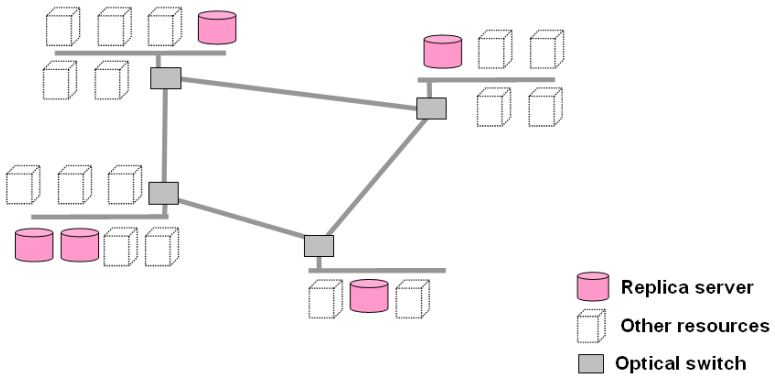


Figure 5-6: Selecting a set of replica servers from the end resource pool of the studied Lambda-Grid configuration

The replica servers were chosen randomly from the end resource pool of each Lambda-Grid configuration as illustrated in Figure 5-6. In order to compare the evaluation results across Lambda-Grid configurations, we used the same ratio of servers to network switches. The number of servers is four times the number of switches and each server has 2 TB of free disk space. Further, for each configuration we generated 5,000 distinct movie objects. We assume these movie contents are of 2K Digital Cinema resolution (up to 2160x1080) with a stream rate of 250 Mbit/s [104]. The average size of these movies is 200 GB. This size runs for approximately one hour and 50 minutes. Table 5-6 shows the number of replica servers and the total number of replicated data objects for each Lambda-Grid configuration.

Table 5-6: Data object replication and distribution of the studied Lambda-Grid configurations

Configuration ID	Number of Switches	Number of End Resources	Number of Replica Servers	Number of Distinct Objects	Number of Replicated Objects
Internet Backbone	1,219	331,203	4,876	5,000	46,159
Brite-1000	1,000	272,476	4,000	5,000	38,908
Brite-2000	2,000	532,771	8,000	5,000	78,243
Brite-4000	4,000	1,069,288	16,000	5,000	156,927
Brite-6000	6,000	1,596,973	24,000	5,000	234,943
Brite-8000	8,000	2,126,462	32,000	5,000	312,424
Brite-10000	10,000	2,769,338	40,000	5,000	389,899

Our decisions for replicating movie objects to replica servers are based on the popularity replication heuristic algorithm [105]. The popularity of data objects follows the Zipf distribution with a skew parameter of 0.75. Using this popularity distribution, each server picked and stored as many objects as storage allowed. The data objects in resource requests for collaborative data visualization and distributed content delivery applications were also picked based on this popularity.

5.1.4 Evaluation Metrics

Using the experimental settings mentioned, two steps are taken in evaluating the three selection algorithms (separate selection (Sep), SA-based combined selection (SA-Com), and hierarchical combined selection (Hier-Com)). For the first set of experiments, key questions include what quality of results each algorithm can achieve and at what cost. We consider two types of application requests – high-performance distributed computing (*ClusterSet*) and collaborative data visualization (*DataViz*), and use the following four metrics.

1. **Selection Cost –** the total running time of the selection process. Good selection results are acceptable and useful only if the selection time is reasonable.
2. **Success Ratio –** the fraction of selections that produce satisfying results. Good selection algorithms should produce a high success ratio even with high application request complexity.
3. **Resource Quality –** the average quality of end resources in the solution, where the quality of resource R is the percentile of R when all resource candidates are sorted according to the user-defined ranking function. Good results produce high resource quality enabling high application performance.
4. **Application Lambda Distance –** the total weighted distance of all lambdas (optical circuits) allocated for each application, or formally $\Sigma_i\, d_i * w_i$ where d_i is the distance and w_i

is the number of allocated wavelengths of the link *i*. Good selection quality will produce low application lambda cost because: 1) it optimizes application communication latency; 2) it minimizes the cost of the ISPs to set up the private network; and 3) it increases the probability of satisfying application requests in the future by conserving use of lambdas.

For the second set of experiments, we evaluated how resource selection decisions of each algorithm affect system throughput, network utilization, and application communication performance in the face of resource contention. Trace-driven simulations and distributed content delivery application workloads were used here.

1. **System Lambda Utilization –** the fraction of available lambdas in the system allocated for use.
2. **System Throughput –** the number of active requests (or applications) in the system. Good network efficiency should be determined by two indicators: high system throughput at high load and a slow growth rate of system lambda utilization at low load.
3. **Application Communication Latency –** the average latency of the allocated circuit paths (lambdas) in the result. Low network latency is crucial for good streaming performance of the distributed content delivery application.

5.2 Selection Quality and Cost

We investigate the significant questions of what solution quality the three selection algorithms (Sep, SA-Com and Hier-Com) can achieve and at what cost. Here, we consider two types of application requests – *ClusterSet* and *DataViz*, and use simulations across a range of Lambda-Grid sizes and resource request complexity. For all experiments, we measured and reported the average value of the selection cost and quality over 120 selection trials.

5.2.1 Impact of Request Complexity

In the first experiment, we evaluate and compare the three algorithms (Sep, SA-Com and Hier-Com) using the Lambda-Grid configuration with the multi-ISP, Internet backbone network, and the application request *DataViz* with varying complexity. To observe the impact of the *clusterSize* parameter of Hier-Com, we use three *clusterSize* values (8, 12 and 16). Figure 5-7(a) compares the selection time of different selection approaches. For all cases, Sep and SA-Com have the fastest and slowest running time. This was expected because while Sep separates the tasks of selecting end resources and networks, SA-Com combines them. This increases the problem's complexity significantly. We also see that the Hier-Com approaches improve the selection time over SA-Com by 75.7-90.8 percent. This improvement is attributed to the hierarchical selection technique which considerably reduces numbers of considered candidates and circuit path computation across the large networks. Among the Hier-Com approaches (Hier-Com-8, Hier-Com-12 and Hier-Com-16), Hier-Com-8's average selection time is slightly longer than that of the other two because Hier-Com-8 recursively clusters the network into four hierarchical levels while Hier-Com-12 and Hier-Com-16 does it into three levels.

Figure 5-7(b) compares the selection success ratio of different selection approaches. Overall, success ratio decreases with higher request complexity, and both SA-Com and Hier-Com always achieve higher success rate than Sep. Sep's success ratio drops quickly because with the separate resource selection, network constraints are not evaluated until end resources are chosen. With a higher request complexity requiring more optical links, there is a higher chance that some network requirements cannot be fulfilled. On the other hand, SA-Com's and Hier-Com's success ratio remains high and close to optimal even with the most complex request (n = 10).

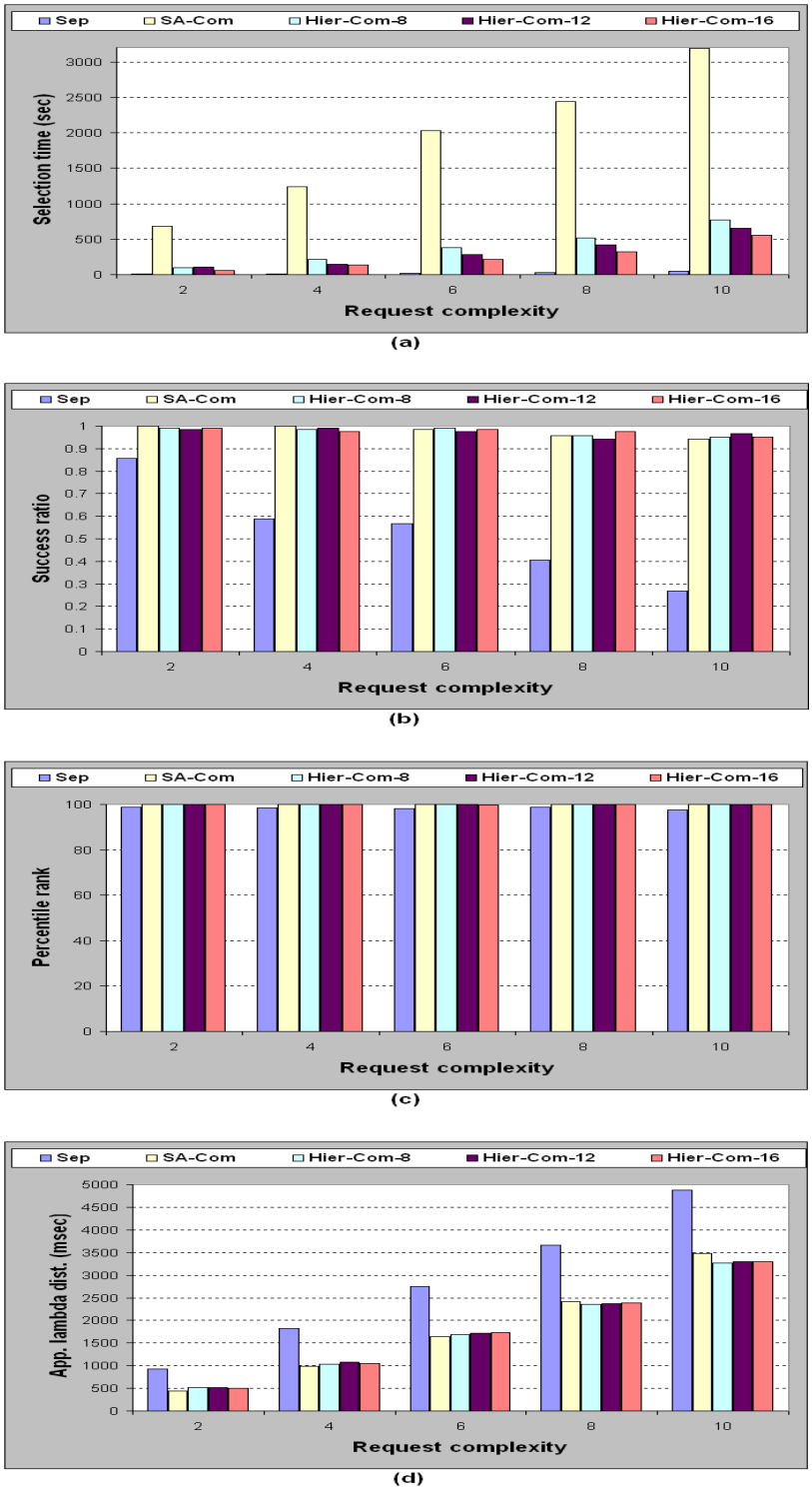


Figure 5-7: Comparison of selection cost and quality with different algorithms using *DataViz(N)* with varying request complexity (*N*): a) selection time; b) success ratio; c) resource quality and d) application lambda distance

The charts in Figure 5-7(c-d) illustrate the resource quality and application lambda cost of different selection approaches. It is apparent that SA-Com and Hier-Com achieve good optimality in both selection quality metrics whereas Sep produces good resource quality but with high lambda cost. This is because in Sep end resources are first chosen based on their end-resource rank regardless of their network proximity. In contrast, SA-Com and Hier-Com provide optimization spanning both communication and end resources. Both SA-Com and Hier-Com have comparable selection quality (within eight percent difference) and improve the lambda cost over Sep by 28.6-52.1 percent. These results show that Heir-Com can be as effective as SA-Com in terms of selection quality, although it scales better with request complexity. In addition, no significant difference was found in the selection quality of Hier-Com using various values of the *clusterSize* parameter for this experiment.

Next, we ran a similar set of experiments using the request *ClusterSet* with varying complexity. The charts in Figure 5-8(a-d) illustrate the resulting selection time, success ratio, resource quality and application lambda cost of different selection approaches. Overall, these results confirm our other findings. Specifically, Sep achieves the fastest running time and high resource quality, though it produces solutions with high network cost. In contrast, both SA-Com and Hier-Com achieve good optimality in both resource quality and application lambda cost. For *ClusterSet*, the saving in runtime gained from using Hier-Com over SA-Com is even more evident, reducing from approximately five hours to five minutes for the largest problem size (ClusterSet(12,8)). There is also a significant improvement in application lambda distance from using SA-Com and Hier-Com over Sep, accounting for 75.1 %.

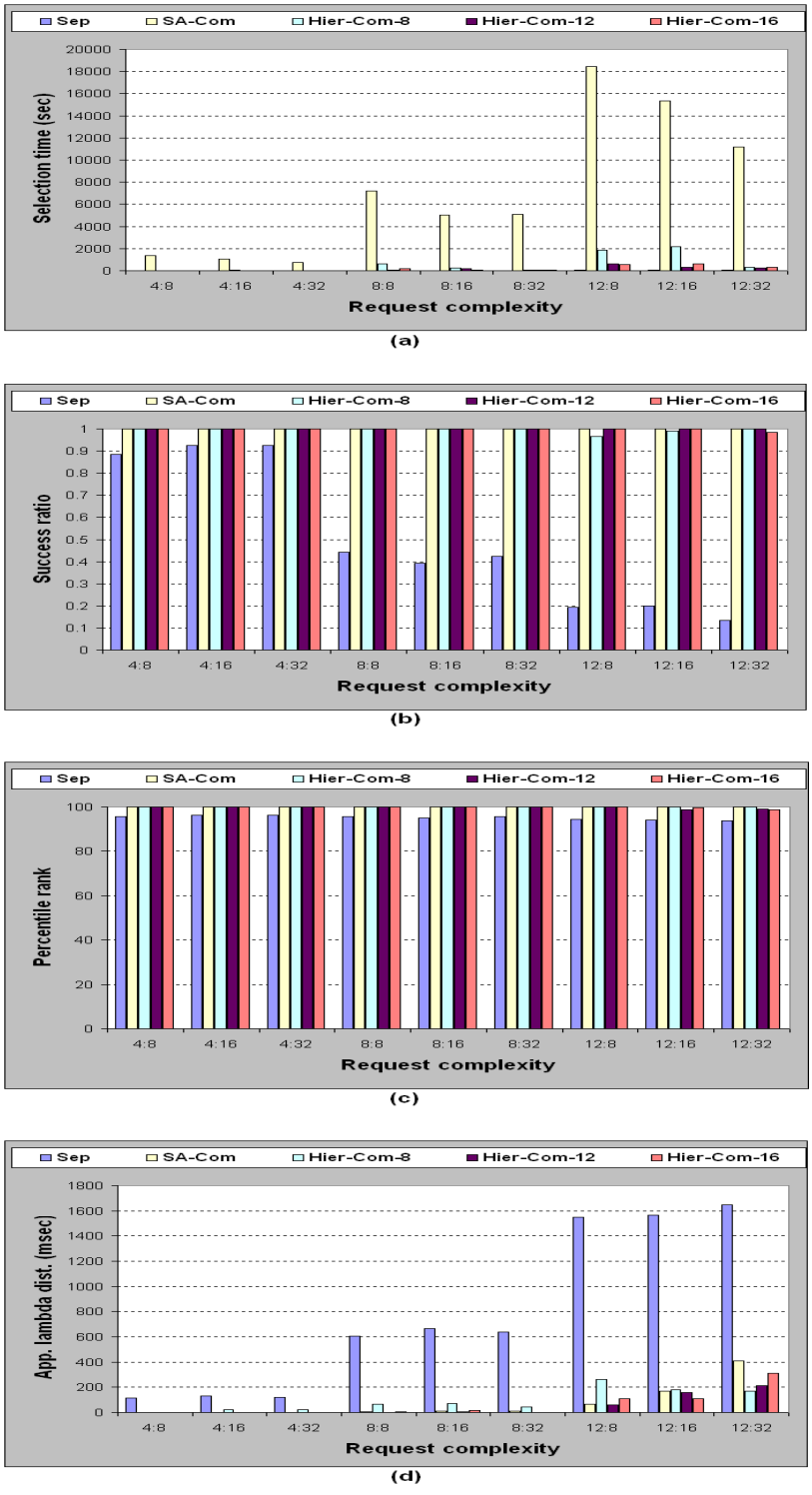


Figure 5-8: Comparison of selection cost and quality with different algorithms using *ClusterSet (C,N)* with varying request complexity (*C, N*): a) selection time; b) success ratio; c) resource quality and d) application lambda distance

5.2.2 Impact of Lambda-Grid Size

The scalability of the three selection algorithms (Sep, SA-Com and Hier-Com) is evaluated using the Lambda-Grid configurations with BRITE topologies with varying numbers of switches and end resources. Our largest resource environment is comprised of 10,000 network switches and 2.7 millions of end resources.

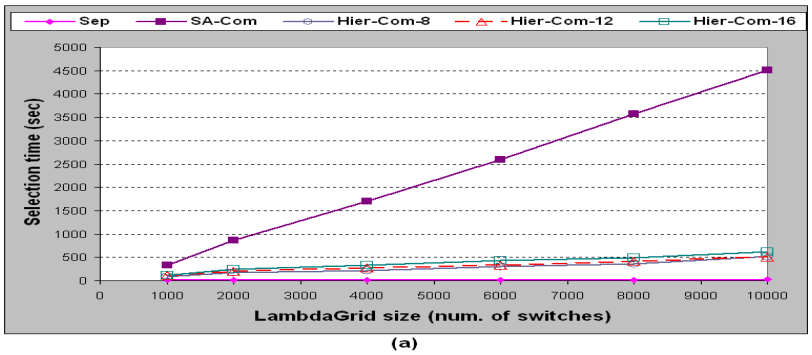


Figure 5-9: Comparison of selection time with different algorithms using DataViz(6) with varying Lambda-Grid size

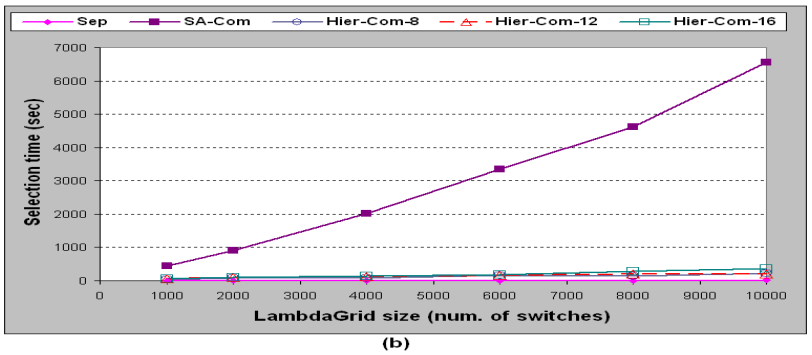


Figure 5-10: Comparison of selection time with different algorithms using ClusterSet(12,16) with varying Lambda-Grid size

The charts in Figure 5-9 and Figure 5-10 illustrate the running time of different selection approaches with varying Lambda-Grid complexity for DataViz(6) and ClusterSet(12,16), respectively. The charts show a similar trend for all algorithms where the selection time increases with larger Lambda-Grid size. It is evident that SA-Com has the

slowest running time and the fastest growth rate due to the complexity of combined selection. While SA-Com offers good selection quality, its high selection cost makes it unacceptable for large resource environments. However, we find that the hierarchical selection technique in Hier-Com reduces the selection cost significantly, making it attractive for large Lambda-Grids comprised of thousands of switches and millions of end resources. Further, among the Hier-Com approaches, the selection time slightly increases with higher *clusterSize* value. This is because using smaller *clusterSize* value results in the network partitioning into larger-size domains at the top hierarchical level and thus larger numbers of bad resource candidates can be filtered out early.

5.3 System Lambda Utilization and Throughput

A key requirement for a resource selection algorithm is that it must enable efficient resource utilization in order to maximize the potential of deploying future applications on a fixed resource environment. To investigate how different algorithms affect network utilization in the presence of resource contention, we used synthetic distributed content delivery workloads and trace-driven simulations. Our workloads are traces of content delivery requests. Requests are removed from a queue individually and each is allocated a private network path to the replica server storing a particular data object. If the request cannot be satisfied (e.g., no available optical path), it will be placed in the system queue and re-evaluated the next time some active requests terminate. In order to observe the system under different loads, we scale the request rate. For each rate, we use 25 traces, each with 52,000 requests shuffled in a random order. For all experiments, we use the Lambda-Grid configuration with the multi-ISP, Internet backbone network.

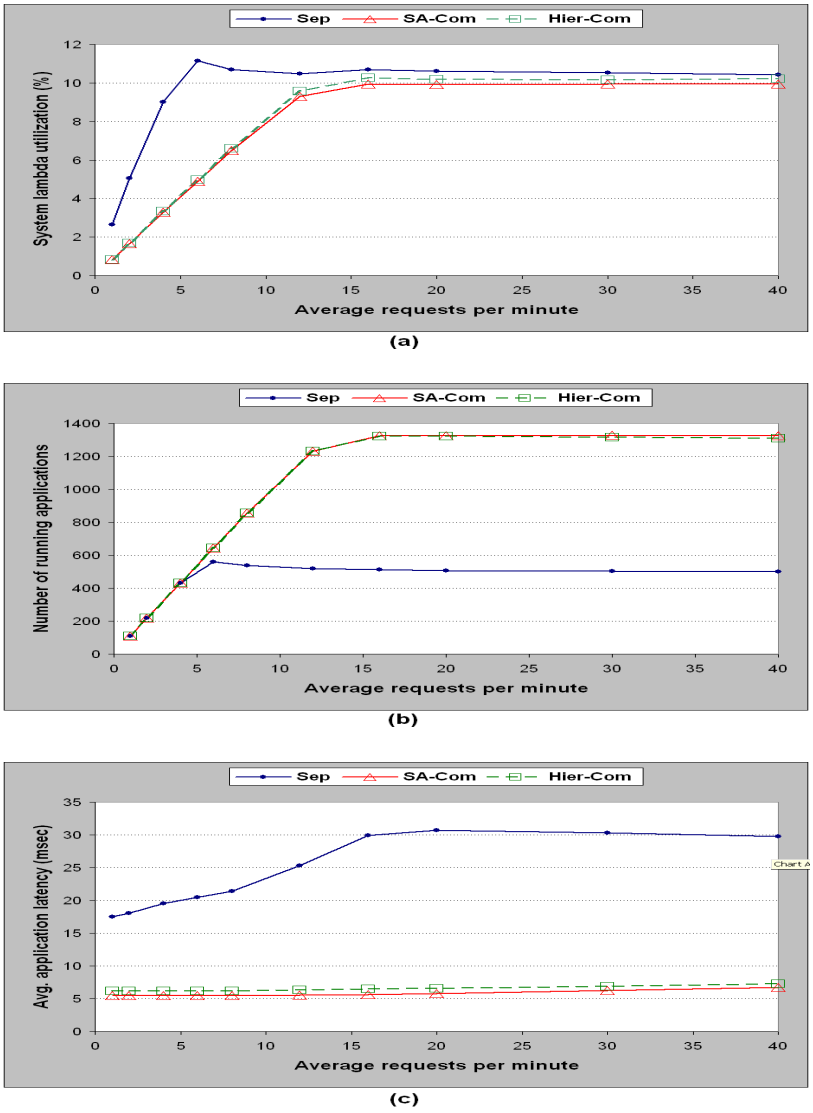


Figure 5-11: Comparison of resource efficiency and application performance of different algorithms as a function of the request rate: a) system lambda utilization; b) system throughput and c) average application communication latency.

Figure 5-11(a) illustrates the average system lambda utilization of the three selection algorithms (Sep, SA-Com and Hier-Com). It's apparent for low system load (<6 requests/min), the utilization grows with higher request rate for all algorithms. This is as expected, because there is little resource contention and all new requests can be allocated with private optical circuits (lambdas). Because Sep makes less efficient use of lambdas – its solutions require longer circuit paths on average, its growth rate is faster than that of SA-Com

and Hier-Com. The growth rates of SA-Com and Hier-Com are comparable. This implies they achieve a similar level of resource efficiency.

At high load when the network becomes congested, the utilization of all the algorithms reaches a saturation point. All algorithms have a comparable saturation point between 9.9 and 10.3 %. This limit is due to the nature of our workload traces that contain many requests requiring lambda paths over some bottleneck links and thus cannot be simultaneously satisfied.

Figure 5-11(b) compares the average system throughput of the three algorithms. At low load, the system throughput for all algorithms is indistinguishable. This is because virtually all new requests can be satisfied and admitted. At high load when the system enters the steady state, Sep's system throughput is the lowest at 499 applications while SA-Com and Hier-Com achieve a comparable throughput at 1,327 and 1,316 applications, respectively. This is because SA-Com and Hier-Com combine the selection of communication and end resources, enabling the system to avoid the selection of bad replica servers to which the clients cannot have good connectivity (or do not have access at all due to the resource contention).

Figure 5-11(c) compares the average application communication latency of the three algorithms. For all cases, SA-Com and Hier-Com achieve a comparable level of application latency. As well, they both outperform Sep. This is as expected, because SA-Com and Hier-Com can choose the replica servers based on their connectivity to the requesting clients, while Sep cannot. Here, SA-Com slightly outperforms Hier-Com because Hier-Com simplifies the problem with recursive network clustering and candidate filtering. However, the increase in average application latency is small, accounting for 0.695 msec on average.

5.4 Summary

In this chapter, we have shown that high application performance and efficient resource usage can be simultaneously achieved by employing specific algorithms for combined resource selection. To solve complex resource selection problems, we presented two novel combined selection algorithms based on simulated annealing (SA-Com) and top-down hierarchical selection heuristics (Hier-Com). As shown through simulations, both algorithms offer good optimality in both end resource quality and network transmission cost, a key requirement for achieving high application performance and resource efficiency. Further, the algorithm based on hierarchical selection (Hier-Com) not only achieves excellent quality of results, but also scales well with both application request complexity and Lambda-Grid size. For instance, it's capable of identifying good solutions within several minutes for large resource environments comprised of thousands of optical switches and millions of end resources.

Chapter 6. Network Information Sharing Challenges and Impacts

Previously, we evaluated different resource selection strategies for realizing application resource requirements in Lambda-Grids. For all experiments, we have assumed that applications and the underlying resource planning service (DVC-RCP) have full knowledge of available network resources, including network topology and link capacity/usage information. While this assumption is applicable for small Lambda-Grid testbeds, it is no longer acceptable for large-scale systems spanning the wide area. Typically, large networks (including the Internet) are partitioned into sub-networks which provide scalability and autonomous administrative domains for each Internet Service Provider (ISP). Because interworking between ISPs raises issues of security, trust, and financial benefits, they are not willing or able to share details of their internal networks [106].

In configurable optical networks, information sharing is crucial for effective path computation across networks and for good resource selection decisions for distributed applications. This chapter will investigate how the available information affects applications' and service providers' ability to utilize network resources. Section 6.1 deals with both key motivations and difficult issues of information sharing. In Section 6.2, we characterize the basic types of information that might be shared between ISPs and provide a spectrum of network information models (NIMs). Section 6.3 describes the methodology used to evaluate the proposed models. In Section 6.4 and Section 6.5, we evaluate the intra-domain and inter-domain impacts of the proposed models on application performance and network efficiency. Lastly, the results are summarized in Section 6.6

6.1 Information Sharing Challenges

While configurable optical networks provide dramatic opportunities for new application capabilities, they also present significant information sharing challenges. Network information (including network topology, link capacity and usage, peering points, etc.) is crucial for effective path computation across networks for distributed applications and efficient traffic engineering for ISPs. This type of traffic management allows the network resources (lambdas) to be utilized more efficiently and leads to better network productivity. However, there are many reasons why ISPs are not inclined or able to share their internal network information.

- **Security** – revealing sensitive details of ISPs' internal networks (e.g. switch locations, core links without backup) makes them vulnerable to a range of security threats, including Denial-of-Service (DoS) attacks [107]. Because such threats potentially cause an infrastructure loss and/or service discontinuity, this information is treated as confidential.
- **Financial Benefits** – publishing internal network information could point a way to other ISPs to gain competitive advantages by offering better network coverage, capacities and/or quality-of-service. This could drive customers elsewhere. In a bandwidth broker model [108], exposing this information also makes ISPs lose bargaining power over selling services to more profitable customers.
- **Internal Network Management** – by advertising detailed internal network information and letting an external entity manage path selection through their networks, ISPs lose control over their resource usage and management. This may cause poorly utilized resources, thus making them unwilling to share information.

- **Interdomain Routing Policy Enforcement** – information hiding provides a means for ISPs to enforce interdomain routing policies. Certain network providers may refuse a transit service to all or a restricted set of other carriers by advertising to their neighboring peers only those routes they use or allow [109].
- **Protocol Heterogeneity** – ISPs may use diverse network management protocols (e.g., PNNI [110], OSPF with GMPLS-extension [46]) to obtain and propagate topology as well as resource information inside their networks. Incompatibility among these protocols may limit the nature and extent of network information that could be shared.

Although detailed network information is important for effective network management, it is often unavailable due to numerous issues of security, economics, and politics. This poses key challenges for controlled information sharing that must not only enable effective path selection for Grid applications, but also maintains competitive advantages for individual ISPs.

6.2 Network Information Models

A critical challenge for configurable optical networks is definition and widespread acceptance of Network Information Model (NIM). This provides information about network capabilities and resources (and possibly in the future, reliability, price, etc.) to higher levels of the system; that information informs the selection and configuration process of a private network for applications. Ideally, NIM would maintain a competitive advantage of individual ISPs, while at the same time enabling effective network resource selection for high application capabilities and resource efficiency.

Here, we describe assumptions on the architecture of a configurable optical network, characterize network information, and define a spectrum of network information models.

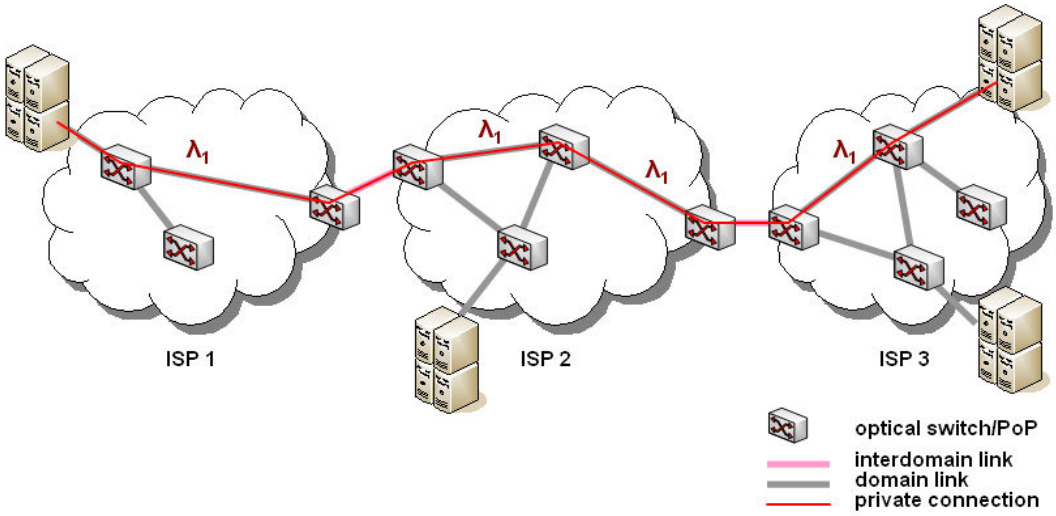


Figure 6-1: Physical architecture of a multi-carrier, optical circuit-switched network

6.2.1 Information Categorization

A configurable optical network consists of a collection of optical switches interconnected by DWDM optical links. In such a network, a connection is created on-demand and formed by a set of optical switches. These switches forward data along the established circuit paths. In today's Internet, networks are partitioned into sub-networks providing autonomous administrative domains for each Internet Service Provider (ISP). These provide the autonomy and scalability of the Internet. Figure 6-1 depicts a simple example of a multi-domain, configurable optical network. The network consists of interconnected groups of optical switches managed independently by three ISPs.

A network switch linking only to other switches within the same domains is called an *internal switch*, while a switch with links to other domains is called a *border switch*. A link between two border switches is called an *interdomain link*, whereas a link terminating at any internal switch is called a *domain link*. Essentially, we can classify network information into two main categories: *domain* and *interdomain* information.

- **Domain network information** – includes topology and link state information of a domain (e.g., each ISP's network). *Domain topology information* specifies the connectivity between nodes and links within a domain, the latency of each domain link, and which end resources are attached to each node. *Domain link state* information specifies the capacity and usage of domain links. A "node" can be generalized as either an optical switch or Point of Presence (PoP) depending on whether the network is switch-level or PoP-level.
- **Interdomain network information** – includes interdomain (domain-to-domain) topology and connectivity information. *Interdomain topology* information specifies interconnection between domains, including their peering points, and the latency, capacity, and usage of each interdomain link. *Interdomain connectivity* information provides network reachability information among ISPs. It can be viewed as "distance vector" information which is similar to that of BGP [109] indicating at which domains and via which interdomain paths a particular domain can be reached.

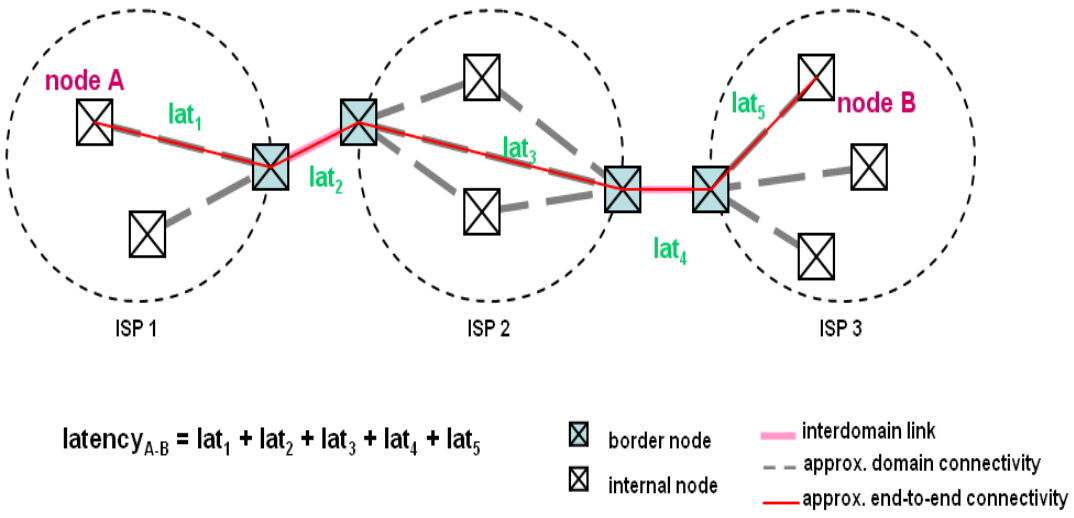


Figure 6-2: Approximating the latency of an end-to-end network path across domains using ConnDom

6.2.2 Model Definition

We present six different network information models below. Figure 6-3 summarizes what types of information are available for each model.

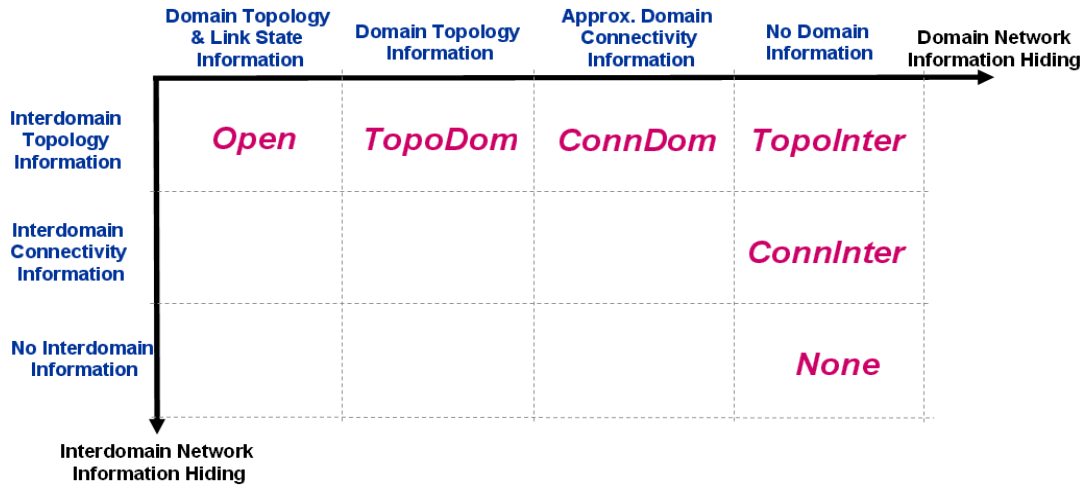


Figure 6-3: Information details of the studied network information models

1. **Open Interdomain, Open Domain (*Open*)** – provides complete domain and interdomain network information. It assumes complete trust amongst ISPs (i.e., an open infrastructure), allowing an external agent to control the selection and configuration process of an entire network path for an application. This simple model is widely deployed in experimental Grid and advanced optical network testbeds, such as OptIPuter [15], CANARIE's CA*net 4 [20], and CHEETAH [18].
2. **Open Interdomain, Topology Domain (*TopoDom*)** – includes all network information except domain link state information. Here, the key idea is while ISPs can profitably share their domain topologies, they aren't willing to reveal information about internal resource capacity/usage. This is because others can exploit it for their own competitive advantages. An exemplary use of this model is an ISP with multi-regional domains (e.g., AT&T US and Europe) which are operated by different business units with their own revenue targets [106].

3. **Open Interdomain, Connectivity Domain (*ConnDom*)** – provides interdomain topology information and an approximation of domain link connectivity. The rationale here is although complete domain information cannot be shared, some abstraction of domain connectivity can be useful to enable more effective path selection [106]. Specifically for our approach, the model provides approximate latency of connectivity between border nodes and between pairs of each border and internal node within a domain. Figure 6-2 gives an approximation of domain connectivity of the network in Figure 6-1. It can be seen how this approach is useful to estimate the total latency of a circuit path across domains, while hiding physical domain topologies. In our implementation, we approximate this latency by computing the latency of the shortest physical network path between two nodes assuming infinite lambdas on links along the path.
4. **Topology Interdomain (*TopoInter*)** – includes interdomain topology information. Due to numerous economics and security issues, each ISP hides all details of its internal network. Although this provides limited domain information, it also offers diverse interdomain paths.
5. **Connectivity Interdomain (*ConnInter*)** – provides interdomain connectivity information. This approach reflects the philosophy of interdomain routing in today's Internet which relies on the Border Gateway Protocol (BGP) [109] to disseminate interdomain reachability and route information. Being distance-vector-based, the model offers neither diverse interdomain path nor link state information.
6. **No Information (*None*)** – doesn't provide any network information. A potential connection between two edge devices can only be inferred from their network interface card (NIC) speed.

6.3 Methodology

Our methodology is described next to evaluate the impact of network information models on applications' and ISPs' ability to utilize network resources. In brief, we use trace-driven simulation across a range of realistic ISPs' metropolitan, national, and global networks, and use the distributed content delivery application (See Section 5.1.2.3) as our workload. In the following, we describe the details of our simulation model, resource selection strategies, and the evaluation metrics.

6.3.1 Lambda-Grid Configuration

To make our evaluation of NIMs broadly useful, we consider a range of realistic ISP optical network topologies. Ideally, the studied networks must be diverse in size, network design, complexity and geographical presence. This will allow us to explore the impact of these factors on the utility of different network information. Unfortunately, due to numerous economics and security issues, ISPs often keep their physical fiber topologies confidential. To evaluate the proposed NIMs, our strategy is to utilize real ISP fiber network topologies wherever possible and also use ISP PoP-level network topologies to approximate their physical fiber maps.

Our metropolitan network topology models were derived from AboveNet's metro-area fiber maps [111]. While a few ISPs publish this information, AboveNet provides the most comprehensive network maps. We chose to use eight of the AboveNet metro-area network topologies (all in major cities) and decoded them manually from the published fiber map images. The details of these networks are summarized in Table 6-1.

Table 6-1: Details of the studied AboveNet's metropolitan network topologies

City	Number of PoPs	Number of Links	Average Edge Degree	Average Link Latency (msec)
Boston	15	19	2.533	0.016
Chicago	33	43	2.606	0.030
Houston	25	34	2.720	0.036
Los Angeles	24	24	2.000	0.022
Philadelphia	22	25	2.273	0.031
San Francisco	38	47	2.474	0.031
Seattle	23	25	2.174	0.030
Washington DC	35	44	2.514	0.028

Our wide-area network topology models were derived from Rocketfuel's router-level, ISP backbone network map collection [93, 94]. Originally, these maps were extracted from the "traceroute" data generated by 300 traceroute web servers across the world. We carefully selected eight ISP network maps – AT&T, Ebone, Exodus, Level3, Sprint, Telstra, Tiscali and Verio; these are large and diverse enough for meaningful study. Using these ISPs, we reduced their router-level topologies to PoP-level (city-level) topologies. Specially, we grouped routers by their geographical locations which were inferred from their DNS names [112]. This reduction simplified our analysis but preserved validity as ISP's traffic engineering decisions are usually made at the PoP-level [114, 115]. The details of these ISP networks are summarized in Table 6-2.

Table 6-2: Details of the studied ISP backbone network topologies

Internet Service Providers (ISPs)	Network Presence	Number of PoPs	Number of Links	Average Edge Degree	Average Link Latency (msec)
AT&T	US	110	140	2.546	1.918
Ebone	US/Europe	27	46	3.407	2.110
Exodus	US/Europe	22	36	3.273	5.672
Level3	Global	48	4000	16.667	6.678
Sprint	Global	44	86	3.909	6.792
Telstra	Australia	55	57	2.073	4.648
Tiscali	Europe	47	80	3.404	2.986
Verio	Global	119	229	3.849	3.633

To study the impact of various inter-domain factors, we used a realistic map of the multi-carrier, Internet backbone network outlined in Section 5.1.1 (see details in Table 5-2).

This network map is comprised of ten large ISPs – AT&T [84], BT [85], Cogent [86], Global Crossing [87], Level3 [88], NTT/Verio [89], Qwest [90], Sprint [91], Time Warner [92], Verizon [79]. While we considered only a small number of ISPs, these ten ISPs are the dominant network service providers in the world (9 tier-1 ISPs and 1 high-degree tier-2 ISP [116]), and together account for a large percentage of today's optical fiber infrastructures. It should be noted that we didn't use Rocketfuel's ISP network topologies (which we used to study intra-domain factors above) because only five of these ISPs are directly peered and inadequate for meaningful study.

Once these network topologies were derived, we assigned lambdas and latency for each link in these networks. For each link we assigned 20 lambdas, each at 1 Gb/s. To obtain the latency of a link between two PoPs, we first determined the latitude and longitude of their geographical presence, calculated their distance using the great circle method [97], and computed the latency using this distance divided by the speed of light. Using this method, we assumed all links are laid along the shortest path between two cities (PoPs).

Using a statistical Grid resource generator [33], we generated end resources (cluster and host information) with the distribution matching the currently deployed Grid infrastructures, such as TeraGrid [11] and iVDGL [12]. These resources were given unique IP addresses and randomly assigned to PoPs of each network topology model. Each PoP consists of 270 end resources on average, and each resource has a 10 Gb/s uplink to the core network.

6.3.2 Application Model

To evaluate the proposed NIMs, our workloads are synthetic traces of movie content delivery requests. Each request is as shown in the application abstract configuration model on the top of Figure 6-4, which specifies a client's IP address (chosen randomly), a requested movie, and a private network path (1 Gb/s). It's assumed each of the derived networks in

Section 6.3.1 contains a set of replica servers, each maintaining a collection of movie contents. For each request, the goal is to find the server with the movie replica closest to the client. The notion of "closest" is determined by the minimum lambda distance (or latency) between the client and the chosen server. If the request cannot be satisfied, it will be placed in the system queue and re-evaluated when some lambdas in the system are released.

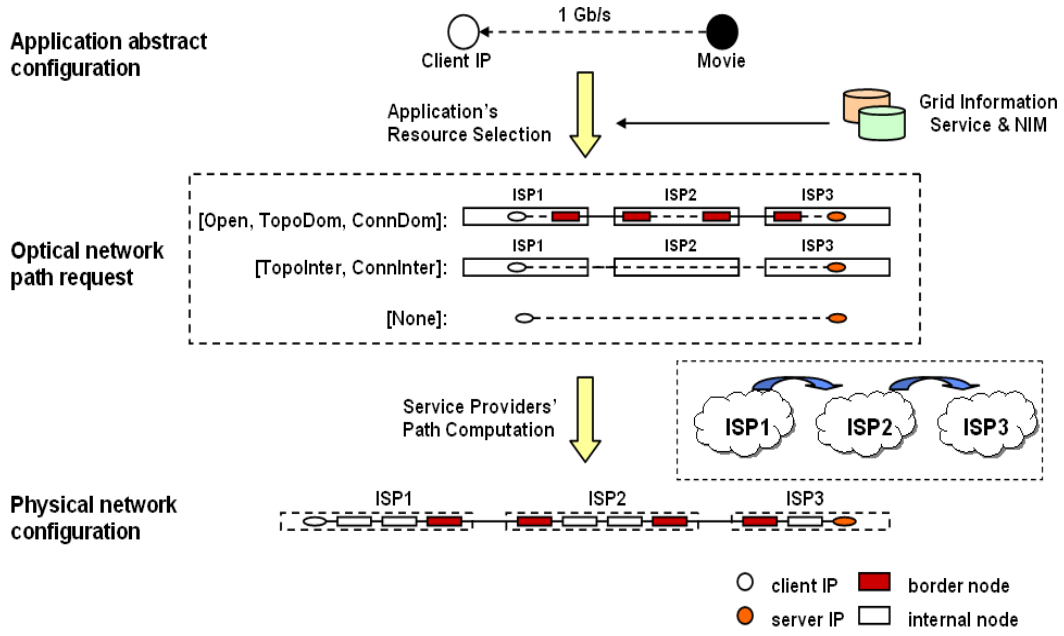


Figure 6-4: Resource selection and network path computation architecture for a distributed content delivery application

The outcome of resource selection for each application request above is an optical path request to network service providers. As shown in the middle of Figure 6-4, resource selection using different NIMs results in three types of optical path requests. For all NIMs, the optical path request specifies the IP address of the client and chosen server together with a "loose" network path between them. For TopoInter and ConnInter, the loose network path is an interdomain path, a result of path computation and selection using interdomain topology and connectivity information. For Open, TopoDom and ConnDom, this path is also decorated

with information about which border nodes should be used to make connections between providers' networks. Such border nodes are derived from the chosen end-to-end network path from resource selection using domain network information provided by Open, TopoDom and ConnDom. For None, the loose network path is merely an abstract link. Subsequently, the optical path request is given to the corresponding ISPs in order. As the request crosses different ISPs, a representative entity of each ISP uses its internal network information to compute a specific intra-domain path through its network. Given these intra-domain paths, a final end-to-end physical network configuration is derived.

For each network model, the locations of replica servers were randomly chosen from its end resource pool. The same ratio of servers to PoPs was used and the number of servers is four times that of PoPs. Each server has 2 TB of disk space. We generated 500 movie objects for each metro-area network model and 5,000 objects for each of the rest. We assume these movie contents are of 2K Digital Cinema resolution with a stream rate of 250 Mbit/s [104]. The average size of these movies is 200 GB, running approximately one hour and 50 minutes. Our decisions for replicating movie objects to replica servers are based on the popularity replication heuristic algorithm [105].

6.3.3 Replica Server Selection and Network Path Computation

To select a replica server with the content replica requested by an application, we use the algorithm shown in Figure 6-5.

Replica Server Selection Algorithm

1. Determine all servers with a replica of the requested movie using information provided by Grid Information Service (GIS)
2. Use the provided NIM to compute (or approximate) the "shortest" path between the client and each server candidate
3. Among all candidates, select the server with the "shortest" path

Figure 6-5: Description of the replica server selection algorithm

This is a greedy search algorithm which leads to good results that are close to optimal. The notion of a "shortest" path varies according to the considered NIM:

- **Open** – uses all network information to determine the minimum-latency, end-to-end path
- **TopoDom** – assumes infinite lambdas on domain links and determines the minimum-latency, end-to-end path
- **ConnDom** – uses summarized domain network connectivity and interdomain topology information to approximate the minimum-latency, end-to-end path. If a network path is within a domain, the latency is zero
- **TopoInter** – uses interdomain topology information to find the path with minimum interdomain hop count.
- **ConnInter** – always uses the provided interdomain path in the interdomain connectivity information.
- **None** – assumes an equal cost for all paths.

Given an optical path request, each ISP computes an intra-domain path through its network. A pair of ISPs may have multiple peering points. Without specific border nodes provided, we implemented the "early exit" peering policy for an upstream ISP to select an intra-domain path to a downstream ISP. Specifically, the upstream ISP uses the peering point closest to the source (or ingress border node) as destination for path computation and selection. In [94], Spring et al. discovered "early exit" is the most common policy accounting for 20-30% of all ISP pairs in the Internet.

6.3.4 Evaluation Metrics

The following four metrics were used to evaluate the proposed information models:

1. **System Lambda Utilization** – the average fraction of available lambdas in the system allocated for use.
2. **System Throughput** – the average number of applications running in the system.
3. **Application Communication Latency** – the average lambda distance (or latency) of the network path that is allocated for each application.
4. **Network Setup Cost** – the average number of optical circuit setups that must be configured for each application.

Note that high system lambda utilization doesn't necessarily mean the network resources are efficiently utilized. The system may experience high network utilization due to uneconomical use of resources. On the other hand, it may experience low resource utilization due to the nature of a workload (e.g., a large number of requests to the same bottleneck links). Good network efficiency should be determined by two indicators: high system throughput at high load and a slow growth rate of system lambda utilization at low load. Good application performance is determined by low application communication latency. To minimize the operational cost of ISPs, low network setup cost is preferential.

6.4 Impact of Intra-domain Factors

In this section, we evaluate the impact of various intra-domain factors on the usefulness of network information models across a range of realistic ISP metropolitan, national and global networks. Here, we consider only the three models (Open, TopoDom and None) to investigate the utility of domain topology and link state information.

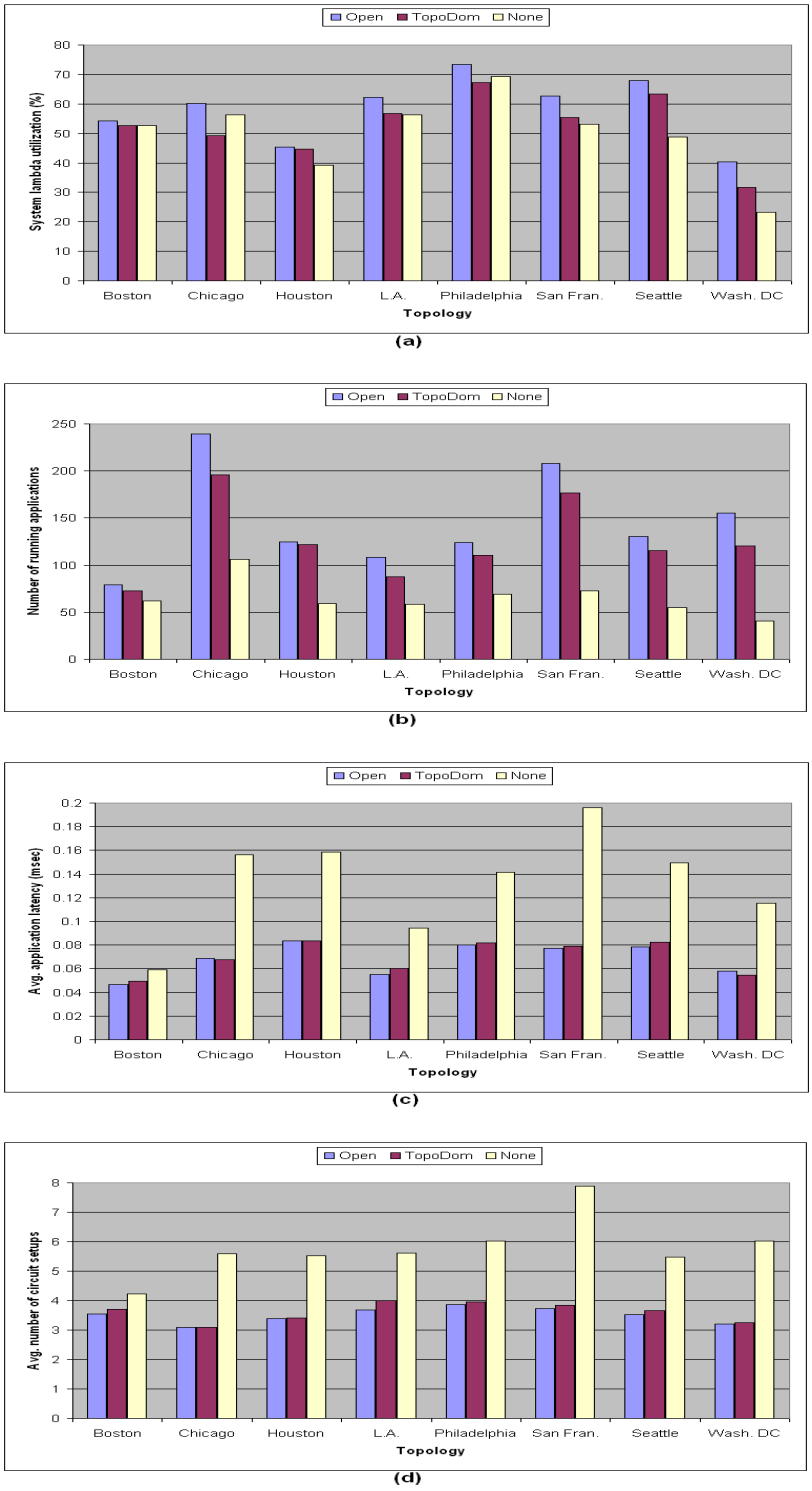


Figure 6-6: Evaluating the intra-domain impact of network information models using metro-area networks: a) system lambda utilization; b) system throughput; c) average application latency; and d) network configuration cost

6.4.1 Metropolitan Network

We used eight AboveNet metropolitan networks (see Table 6-1) to evaluate and compare between the three NIMs (Open, TopoDom and None). The results below were reported using the request rate of 10 requests/min, which is high enough for all metrics to be measured at their saturation regions.

Figure 6-6(a) shows the average system lambda utilization for Open, TopoDom and None. For all topologies Open achieves the highest lambda utilization. This is because Open provides complete domain information allowing us to identify good solutions with low-latency paths and avoid congested links at high system load. Lambdas are more efficiently used as a result and more applications admit into the system on average. In terms of lambda utilization, we see no clear superiority between TopoDom and None. As explained below, while TopoDom produces higher system throughput, it allocates fewer lambdas per application on average.

As shown in Figure 6-6(b), for all topologies Open's system throughput is always higher than that of TopoDom, and both outperform None. These results indicate that domain topology information is a key for achieving good system throughput, while link state information has a positive impact as well. Without domain topology information, a lot of solutions with long-latency paths are chosen but they cannot be simultaneously realized due to their high demand of lambdas. Another finding is the size and topological structures of a metro-area network have impact on the advantage of Open over TopoDom. The superiority of Open becomes more evident in larger and denser networks, such as Chicago and Washington DC. This is because these networks offer more diverse paths, and link state information can be used to take advantage of these paths when the network becomes congested.

The charts in Figure 6-6(c-d) respectively illustrate the average application communication latency and network setup cost for the three models. We can see the comparable results between Open and TopoDom, while they both achieve much lower application latency and network setup cost than None. This shows the significance of domain topology information for these performance dimensions, while link state information has little impact. Domain topology information is required for computing the latency of a network path which is essential for comparing the quality of solutions and making efficient path selection decisions. We also see the correlation between the application latency and network setup cost metrics because a NIM that makes more efficient use of lambdas will likely allocate shorter optical paths and requires less numbers of circuit setups.

6.4.2 ISP Backbone Network

Next, we analyze the utility of the three network information models (Open, TopoDom, None) using real ISP backbone networks (see Table 6-2). In the following results, we used the request rate of 40 requests/min, which is high enough for all metrics to be measured at their saturation regions.

The chart in Figure 6-7(a) illustrates the average system lambda utilization for the three models. We find that Open always achieves the highest utilization. Depending on ISPs, Open outperforms TopoDom and None by 3.1-9.8 percent and 2.8-21.0 percent, respectively. As explained above, this is attributed to domain topology and link state information which leads to better overall network efficiency and system throughput.

The simulation results show a strong influence of network topology design on system lambda utilization. The most common network design among the studied ISPs is “hub-and-spoke”. These ISPs, including AT&T, Telstra, Tiscali and Verio, have stubs in major cities and spokes that fan out connections to smaller cities. For such ISPs, the bottlenecks are the

links connecting major hubs and we observe their system lambda utilization using Open in the range of 22.2 to 36.5 percent. In comparison, Level3 includes PoPs in major cities in US and Europe. While these PoPs are highly connected, the bottlenecks are the links across the two continents. Because our workloads contain a fair number of requests to these links (applications and data replica are in different continents), other links are relatively underutilized. This leads to low system lambda utilization even with Open (13.9 percent). Exodus and Ebone represent another network design paradigm where their network topologies are more balanced graphs – where links are more evenly distributed among nodes. These two networks have no true bottleneck link, and we can achieve higher system utilization for all the three models (48-60 %).

As shown in Figure 6-7(b), for all ISPs, Open achieves higher system throughput than TopoDom, while both outperform None. These results confirm our findings presented above that both domain topology and link state information contribute to better system throughput. Depending on the studied ISPs, we see varying degrees of advantage of Open over TopoDom. This implies the impact of network topology of an ISP on the utility of domain link state information. Among the studied ISPs, we see it is most beneficial using Level3. Figure 6-8 illustrates the portion of the Level3 network in the Northeastern USA. While most cities are highly-connected, Newark and Garden City each have one link to New York. Because these two links have one of the lower latency (0.047 and 0.103 msec), they are highly utilized (often picked by our resource selector). When these links become congested at high load, if the resource selector doesn't know about the link usage, it will continue to choose solutions including these links and fail. With link state information, it can avoid these congested links and select other feasible candidates. While observing similar effects across ISPs, we find its most impact on Level3 because their PoPs are highly-connected and hence a majority of their links are blocked form being utilized.

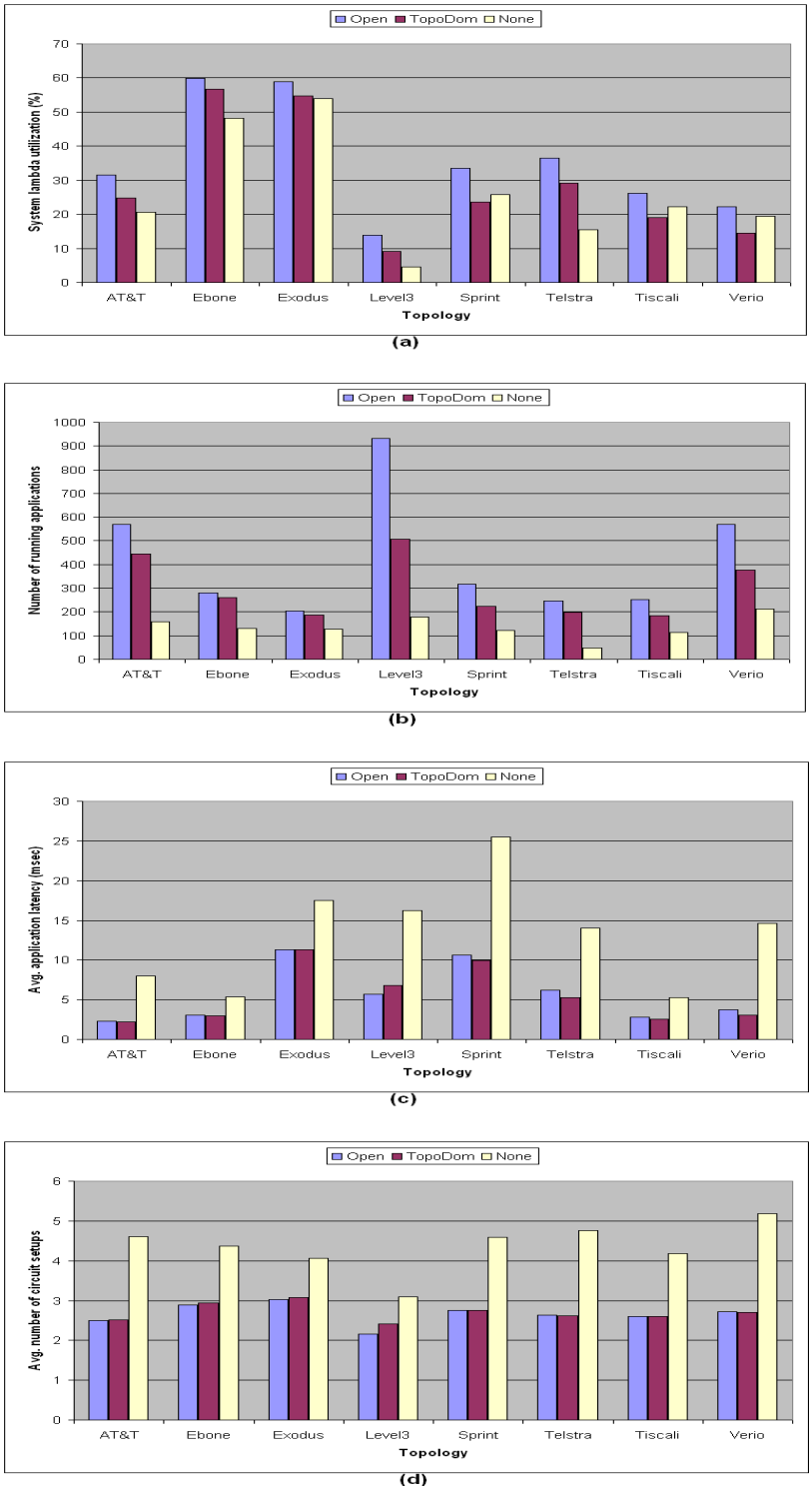


Figure 6-7: Evaluating the intra-domain impact of network information models using ISP backbone networks: a) system lambda utilization; b) system throughput; c) average application latency; and d) network configuration cost

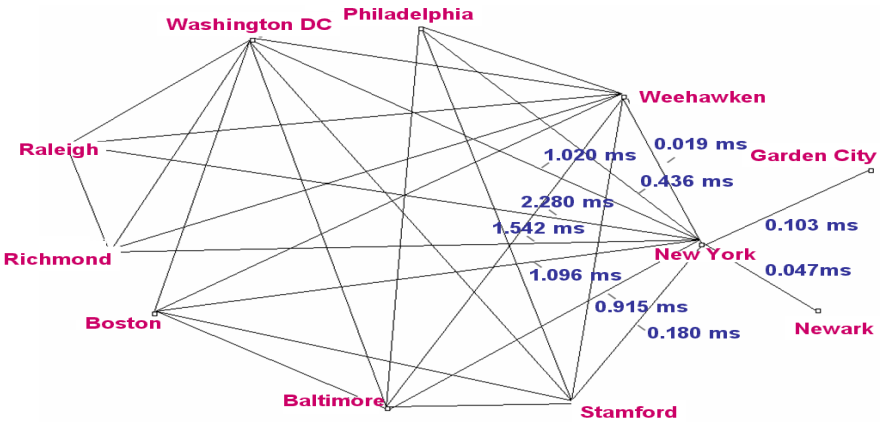


Figure 6-8: Fiber map of the Level3 backbone network in Northeastern USA

Figure 6-7(c-d) shows the comparison of average application latency and network configuration cost for the three models. We see no major difference between the results of Open and TopoDom, while both achieve much lower average application latency and network setup cost than None. These results also confirm our findings presented above that while domain topology information contributes to lower application latency and network setup cost, link state information has little impact on these dimensions of performance.

6.5 Impact of Inter-domain Factors

We next analyze the impact of various inter-domain factors on the usefulness of network information models using the realistic multi-ISP, global network (see Table 5-2). We consider all the six network information models (Open, TopoDom, ConnDom, TopoInter, ConnInter and None) to investigate the utility of different interdomain and domain network information. To observe the system under different loads, we report the results with varying application request rates.

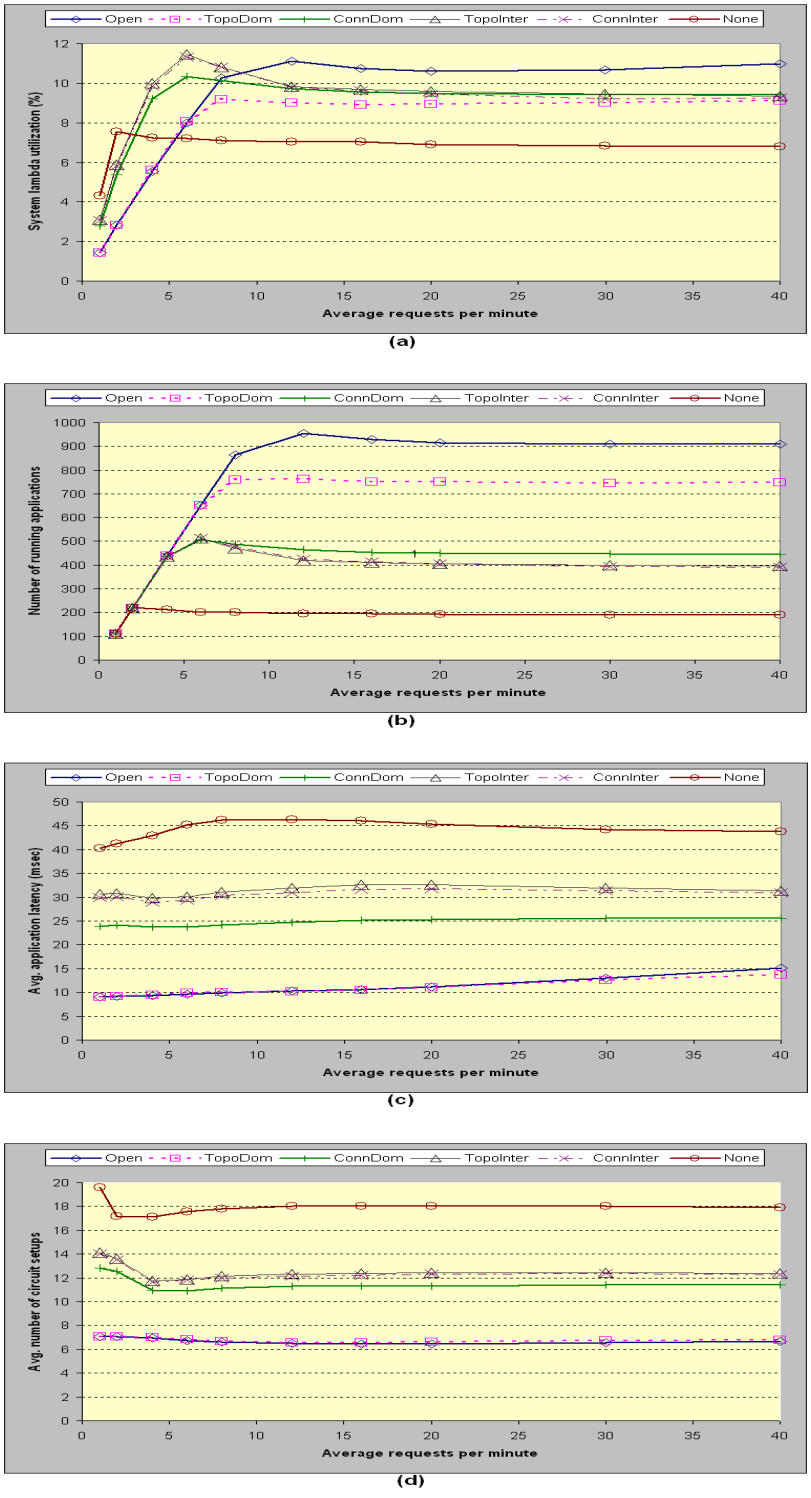


Figure 6-9: Evaluating the impact of network information models using a multi-domain network with top-tier ISPs: a) system lambda utilization; b) system throughput; c) average application latency; and d) network configuration cost

Figure 6-9(a) compares average system lambda utilization for the six information models as a function of a load. At low system load a model with better network efficiency can be observed by its slower growth rate of the utilization with higher load. We see the growth rate of Open and TopoDom is lower than of ConnDom, TopoInter, ConnInter, whereas the growth rate of all these models is slower than that of None. The differences in the demonstrated system lambda utilization are attributed to the network information with varying degrees of abstraction supplied by individual models. These results demonstrate that while interdomain topology or connectivity information alone contributes to better resource efficiency (lower growth rate), domain topology information has the most positive impact on it.

As the request rate continues to increase and the network resources become congested, we see the growth rate of each model moves from linear increase to a flattened saturation region. Open's saturation region is the highest at ~10.7 percent, that of None is the lowest at ~6.8 percent, and the rest have the saturation region in the range of 9.1-9.2 percent. These results show the benefit of domain link state information on improving overall system lambda utilization. At high system load, this information helps to avoid congested links and take advantage of diverse domain paths. Note that none of the studied models achieve lambda utilization close to the full network capacity (only 6.8-10.7 percent). This low network utilization is attributed to the nature of our workload traces that contain many requests requiring long lambda paths over some bottleneck links and thus cannot be simultaneously satisfied.

The chart in Figure 6-9(b) shows the comparison of system throughput for the proposed models. At low load, and for all models, the average number of running applications increases linearly with higher load and at approximately the same rate. This is due to the fact that the network is not congested, and almost all new application requests can be satisfied and

admitted into the system. We find that at high load (>20 requests/min) the system throughput for each model reaches a saturation point. The saturation points for Open, TopoDom, ConnDom, TopoInter, ConnInter and None are at 910, 746, 446, 396, 391 and 190 applications, respectively. These results show that while interdomain topology and connectivity information is useful, domain topology and link state information has greater impact on improving system throughput. Closer investigation reveals that because the studied network contains a small number of large ISPs, we can see more influence of domain information over interdomain information. In addition, since most pairs of ISPs are directly peered in the studied network, we see little benefit of interdomain topology over interdomain connectivity information. While topology information offers diverse interdomain paths, these paths are more difficult to be realized because they span multiple large networks. Lastly we see some advantage of ConnDom over TopoInter, implying the usefulness of approximate domain connectivity information in ConnDom.

As shown in Figure 6-9(c), there are observable differences in the average application latency achieved by different network information models. We find that Open and TopoDom produce the lowest and comparable application latency. This confirms our findings that while domain topology information plays a key role in achieving low application latency, link state information has minimal impact on it. We also find that TopoInter's and ConnInter's average application latency is much lower than that of None. This implies interdomain topology and connectivity information is useful. However, the latency of TopoInter is slightly higher than that of ConnInter. This is because ConnInter limits us to use only the shortest interdomain paths (given in the interdomain connectivity information) thereby leading to lower application latency on average. Lastly, in terms of average application latency, we see some benefit of summarized domain connectivity information in ConnDom. With this information, we can

estimate the latency of a network path across domains. This is useful for comparing the quality of solution candidates and making better path selection decisions.

Figure 6-9(d) shows the average number of circuit setups for different network information models as a function of a load. For all load, both Open and TopoDom require the lowest network setup cost, while ConnDom, TopoInter and ConnInter all produce a higher and comparable cost. These results imply the need for domain topology information to achieve low network setup cost. Interdomain network information also has some impact .

6.6 Summary

In this chapter, we define a spectrum of network information models (NIMs) and evaluate their impact on applications' and service providers' ability to utilize network resources in Lambda-Grids. Our simulation studies show that the choice of model is important, leading to significant differences in system throughput, lambda utilization, network setup cost and attained application performance.

Table 6-3: Summary of utility of different network information on the studied metrics

Network Information	**Usefulness on the Metric**			
	System Utilization	**System Throughput**	**Application Latency**	**Network Setup Cost**
Interdomain connectivity	Medium	Medium	Medium	Medium
Interdomain topology	No	Minimal	No	No
Approx. domain connectivity	Low	Low	Medium	Low
Domain topology	-	High	High	High
Domain link state	Medium	Medium	No	Minimal

Table 6-3 summarizes the utility of different network information on the studied metrics. The usefulness of individual information is determined by its improvement over the previous information factor for a given metric. The results show two significant factors are domain topology and link state information. Domain topology information enables efficient

network path computation and selection, a key driver for high system throughput, low application latency, and network configuration cost. Conversely, domain link state information is useful when a network becomes congested, necessitating the ability to identify and avoid highly-utilized links. At high load, domain link state information improves both system throughput and overall lambda utilization.

Another important factor contributing to better system throughput, application latency, and network setup cost is interdomain connectivity information. When domain information cannot be shared, such connectivity information is useful to approximate the quality of network paths by their interdomain hop count. Another finding is interdomain topology information provides minimal improvement (or sometimes even negative impact) over connectivity information. This is because our studied network has a small number of large ISPs (some with >280 PoPs) and the efficiency of path selection is highly influenced by intra-domain factors. The benefits of interdomain topology information could be more evident in the network with a larger number of smaller ISPs. Lastly, when combined with interdomain topology information, approximate domain connectivity information can be useful. Such information enables an estimation of a network path cost with higher degree of precision, leading to better application latency and system throughput.

Our results show a strong influence of the network topology of an ISP on system throughput and utilization of lambdas. In the network with highly connected nodes (e.g., Level3 and Verio), domain link state information is most advantageous, allowing available lambdas on diverse intra-domain paths to be identified and improving system throughput. Among the studied ISPs, Exodus and Ebone have their links more evenly distributed among nodes and achieve the highest lambda utilization. The ISPs with the “hub-and-spoke” network topology design (AT&T, Tiscali, etc.) have bottleneck links connecting between major hubs,

while Level3 have those crossing between the US and Europe. Due to these links, they achieve lower system lambda utilization.

In short, our results encourage cooperation between ISPs to share internal network information. Such information sharing can make major difference in better resource efficiency and lower operational cost for ISPs and better network service for applications. In addition, part of ISPs' internal network information was already obtained as shown in this research.

6.7 Acknowledgement

Chapter 6, in part, is published as "Evaluating Network Information Models on Resource Efficiency and Application Performance in Lambda-Grids" by Nut Taesombut and Andrew A. Chien in the proceedings of ACM/IEEE International Conference on High Performance Computing and Communication (SC'07), November 2007. The dissertation author was the primary researcher and co-author of this paper.

Chapter 7. Case Studies with Geosciences Applications

The unique capabilities of the DVC include a simple use model for applications, resource configuration optimization, dynamic network configuration, and simple resource naming and communication interfaces that integrate a wealth of underlying network complexity. In this chapter, we present case studies of collaborative visualization environments for earth sciences to demonstrate the DVC capabilities in practice. We demonstrate such collaborative environments can be effectively and conveniently constructed on an international-scale Lambda-Grid testbed; these applications are not feasible without the DVC due to the complexity of heterogeneous, wide-area distributed resource environments and configurable optical networks.

7.1 Collaborative Data Visualization for Earth Sciences

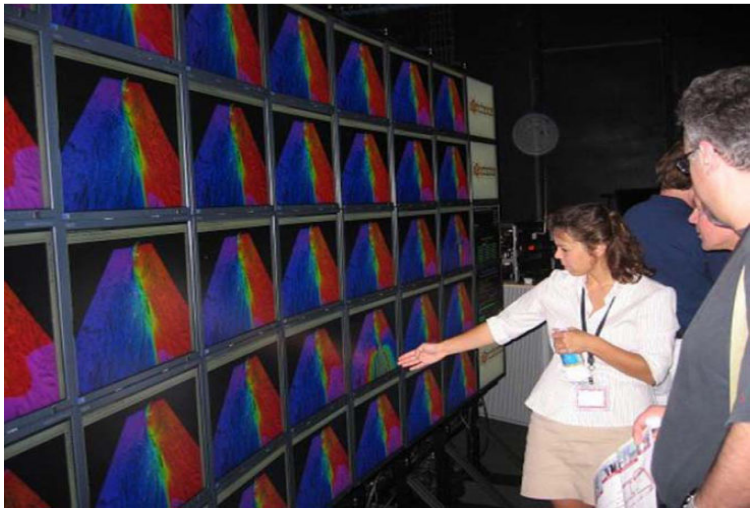


Figure 7-1: Parallel visualization of multiple 3D theoretical models of deformation along the San Andreas Fault in California

The quality and amount of geosciences data being produced, collected, and used in the last few years has risen dramatically. For example, the EarthScope [4] is a National Science Foundation (NSF)-supported project to develop a national cyberinfrastructure to study the

development of the Earth's crusts in North America. The EarthScope's data collections are massive – individual modern 3D seismic images of the Earth's substructures can be as large as 50 GB, and the total seismic data assembled per year exceeds 40 TB [117]. Exploiting the availability of these high-quality images, a wide range of research is being pursued to develop advanced visualization tools [118] that will enable scientists to interactively explore data objects at very high resolution in multiple dimensions. As illustrated in Figure 7-1, scientists are employing parallel visualization of multiple 3D theoretical models to study the deformation along the San Andreas Fault in California.

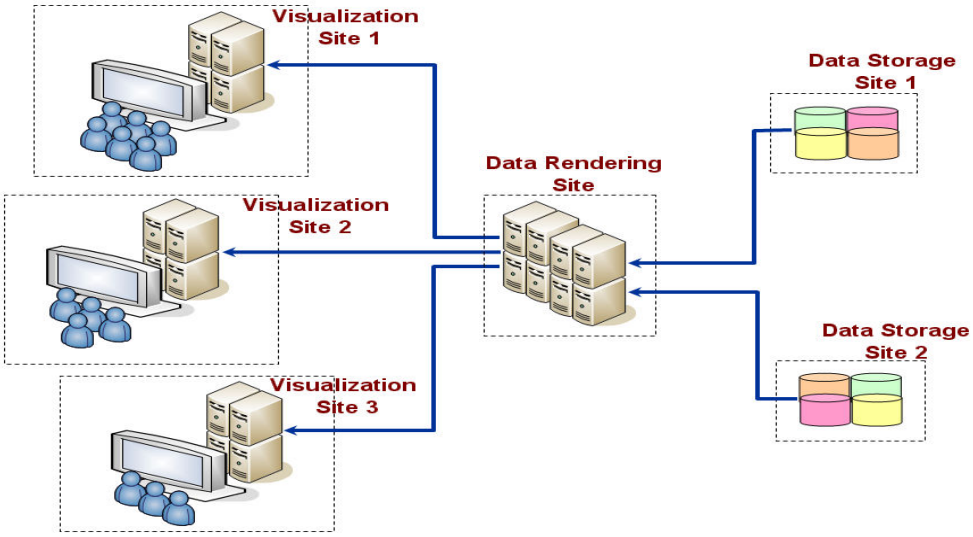


Figure 7-2: Collaborative and remote data visualization system architecture

Collaborative and remote data visualization has become increasingly prevalent and important for geosciences [3]. As shown in Figure 7-2, such collaboration enables researchers from geographically dispersed locations to simultaneously and interactively visualize, explore, and analyze very large data objects in real-time. This improves the quality of scientific data interpretation and the understanding of complex geological systems. However, the development of these applications is facing a challenge in their high demand for network

bandwidth, quality of service, and large-scale resource aggregation across organizations. With today's networking infrastructures (i.e., the Internet), it's impossible for scientists to transfer data quickly to support real-time analysis and achieve good interactive visualization performance.

The ability to build wide-area collaborative visualization environments is made possible with advanced configurable optical networks and resource sharing in the form of Grid [9]. Although there are many previous efforts [119-121] on building such applications on Lambda-Grids, effectively all of existing systems are static or limited in capabilities. Specifically, either their construction requires direct cooperation among IT administrators of the participating organizations or their configurations are fixed with certain sets of physical resources. When users move from one resource configuration to another (e.g., to use different sets of data replica servers), it usually requires extra efforts for application reprogramming or resource reconfiguration.

7.2 Problem, Challenges and Approach

7.2.1 Problem

Using visualization of high-resolution images to study complex geological systems has become very popular among scientists who study the earth. As an example shown in Figure 7-1, researchers at the Scripps Institution of Oceanography (SIO) are using the EarthScope's seismic data to analyze the deformation along the San Andreas Fault zone in California. To produce 3D images of the strain fields resulting from this deformation, they simulate the theoretical models and employ the 'Fledermaus' visualization package [122] to arrange seismic data into a georeferenced coordinate system. The result is a set of 'scene' files that can be viewed and explored by the 'iView3d' visualization tool [123]. To share these

scene flies with colleagues, the SIO researchers create replicas and make them available on a distributed set of storage servers for download. Until recently, such scientific data sharing techniques are confined to local-area environments because a vast number of large scene files are required for meaningful data correlation analysis.

Remote and collaborative data visualization helps enhance the productivity of scientific data interpretation [3]. As illustrated in Figure 7-2, such collaboration enables scientists to interactively visualize and collaboratively analyze the scene files (3D strain field images) with other scientists who are far away. To enable this collaboration, underlying resource infrastructures must support remote resource access across organizational boundaries and high-quality network service. This is made possible with Lambda-Grids, which allow widely dispersed resources to be securely shared through a Virtual Organization (VO) and tightly interconnected with dedicated, high-speed optical circuits.

7.2.2 Challenges

Although the resource requirements of wide-area collaborative visualization environments can be met by emerging Lambda-Grid infrastructures, building these applications remains difficult for many reasons.

First, identifying and selecting network, storage, and visualization resources in the system requires a good understanding of the complex telecommunication and wide-area distributed resource infrastructures. For example, the scene files are replicated and distributed across a large collection of distributed storage servers. It is important to locate the servers with the files of interest that are close to the visualization resources. This will achieve good interactive visualization performance. Further, the resource selection process is not simply a one-to-one mapping between application components and physical resources. Utilizing configurable networks requires the ability to compose communication resources (e.g., optical

links and switches) into end-to-end network connections. Such network composition is computationally hard and requires an understanding of the details involved in underlying network infrastructures.

Second, employing resources in federated systems and wide-area configurable networks involves management of cross-organization security, heterogeneous resource capabilities, as well as multi-domain optical routing and signaling. This requires negotiation with several distinct server providers which impose diverse policies and mechanisms on their resource use. This configuration process can be complex and time-consuming.

Third, collaborative visualization environments employ Grid resources with heterogeneous naming mechanisms. Typically, resources are hidden under firewalls [124], network address translation (NAT) [125] and/or non-routed networks; therefore, their IP addresses can be either private or dynamically assigned. Managing heterogeneous and dynamic resource names (or IP addresses) complicates application development. Further, it may require modification to the application when different sets of physical resources are used due to their assorted names.

Forth, achieving the performance of high speed, long-distance connections requires the use of novel, exotic transport protocols (including UDT, GTP, etc.). Collaborative visualization applications can exploit a mixture of these protocols to optimize data transfers depending upon certain network conditions (e.g., dedicated vs. shared networks) and communication patterns (e.g., point-to-point vs. data aggregation). However, utilizing various protocols with diverse interfaces and implementation complicates application programming.

7.2.3 Approach

To address these challenges, our work employs the DVC to construct collaborative visualization environments. The DVC automates on-demand resource discovery, selection,

and allocation, allowing the applications to be optimally constructed on a high-quality set of resources (e.g., low network latency). A virtual computing environment is created and assigned the allocated resources with virtual names and IP addresses, enabling the applications to be flexibly run on different sets of physical resources without modification. On the created DVC environment, we built a simple collaborative visualization program that acquires scene files on-demand from the remote storage resources and simultaneously visualizes them at distributed visualization sites. This program was implemented with the DVC unified communication interface, which allows it to switch to use different transport protocols without any reprogramming effort.

We measured and evaluated the performance of the collaborative visualization environment establishment. We show the collaborative environment can be quickly constructed in seconds and independently run across different physical resource configurations.

7.3 Experimental Setup

We deploy the DVC prototype on the OptIPuter's international testbed [15] and use it to develop collaborative visualization applications for geosciences. Our experiments include online resource discovery and selection, dynamic resource and network allocation, virtual computing environments, high-speed data transfer, remote visualization and scientific collaboration.

7.3.1 OptIPuter Lambda-Grid Infrastructure

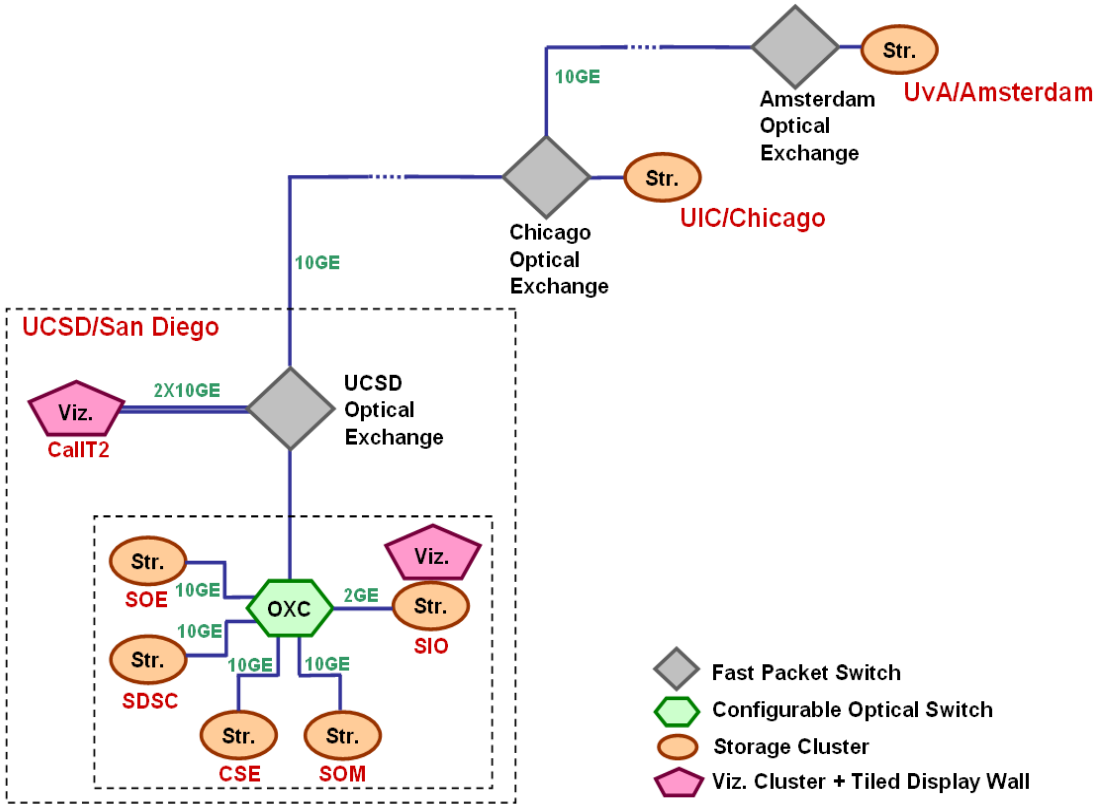


Figure 7-3: The OptIPuter's international Lambda-Grid testbed and iGrid2005 networking infrastructure

Our experiments are constructed on the OptIPuter's international Lambda-Grid testbed [15] and the networking infrastructure provided by iGrid2005 [126]. The OptIPuter [15] is an NSF funded research project exploiting the availability of dynamic high-speed optical paths to provide revolutionary capabilities for emerging e-science. As illustrated in Figure 7-3, the infrastructure is comprised of distributed storage clusters across sites at the University of California at San Diego (UCSD), University of Illinois at Chicago (UIC) and University of Amsterdam (UvA). There are five storage clusters on the UCSD campus, each at the Department of Computer Science and Engineering (CSE), School of Engineering (SOE), San Diego Supercomputing Center (SDSC), School of Medicine (SOM) and Scripps

Institution of Oceanography (SIO). Two visualization clusters are located there as well: one at the CalIT2/UCSD and another at SIO/UCSD. Each of these is connected to a tiled display wall which can visualize multiple data objects in parallel on multiple screens. Each storage and visualization node has a 2.0 GHz or faster, 1 GB of more memory and a Gigabit Ethernet NIC.

The storage and visualization clusters are interconnected by 10 Gbps heterogeneous optical networks. The visualization cluster at CalIT2/UCSD has two 10 Gbps uplink interfaces – 10 Gbps aggregate connectivity to/from UvA and UIC, and 10 Gbps aggregate connectivity to other sites at UCSD. The connection between CalIT2 and UvA/UIC is static and already in place; therefore, no dynamic network setup is needed. This is due to the fact we lack administrative access to the optical switches at Chicago and Amsterdam. The UCSD OptIPuter network is controlled by a configurable optical cross-connect (OXC) switch. This switch, in turn, is controlled by software and capable of dynamically switching connection from one storage cluster on the UCSD campus to the CalIT2 visualization cluster. Each UCSD storage cluster (expect SIO) has one 10 Gbps uplink interface to this switch.

7.3.2 Software Configuration

A set of software components were deployed to set up our experiments. Specifically, we established a Virtual Organization (VO) [10] including all the sites above. We used Grid Security Infrastructure (GSI) as standard security mechanisms for authentication and authorization. We configured and set up Globus GRAM [37] and MDS [35] servers on individual storage and visualization nodes. The former serves as a gatekeeper that authorizes the user for secure remote resource access, while the latter monitors resource performance and creates a directory service for resource discovery and selection. We also installed the iView3d visualization package on each node at the visualization clusters.

Additionally, we set up PIN/PDC servers on three dedicated hosts to manage network resources at UCSD, UIC and UvA. These three sites were organized into three separate network domains. We used the PIN/PDC server at UCSD for both controlling the OXC switch and interdomain routing. However, we used the PIN/PDC servers at UIC and UvA only for interdomain routing because we lack administrative access to control the switches at both sites.

7.4 Experiment Results

We used the DVC prototype to construct a collaborative visualization application on the OptIPuter infrastructure. The application inputted the name of datasets (collections of scene files), acquired them from remote storage clusters, and visualized them with iView3d on the tiled display wall at CalIT2. Simultaneously, the application visualized the same datasets at the tiled wall at SIO; this was done in order to enable the scientists at both sites to collaborate and analyze the data. However, at this site the entire library of the datasets were already present on disks and the requested scene files were loaded locally. We didn't acquire the datasets remotely here because of insufficient incoming network bandwidth to SIO.

To simplify the development of the collaborative visualization application, we created a DVC virtual computing environment that included three storage clusters, two visualization clusters and lambda connections. A virtual namespace was created and assigned individual storage and visualization nodes with unique virtual IP addresses. We implemented the application to utilize these addresses so that it can be run independently on different sets of physical resources.

Our implementation of the collaborative visualization application was written in 703 lines of C/C++ code. Table 7-1 breaks down the number of lines by modules. Without the

DVC prototype, developing the collaborative visualization application with the same complexity would be impractical or it requires manual configuration by IT administrators.

Table 7-1: Number of code lines of the studied collaborative visualization application by modules

Module Name	Number of Lines	Percentage
Global variable declaration	68	9.67 %
Main	98	13.94 %
Resource binding and network configuration	25	3.55 %
Remote data transfer	107	15.22 %
Display device configuration and visualization software launcher	86	12.23 %
Other utility functions	319	45.38 %
Total	703	

7.4.1 Resource Selection and Allocation Performance

```
(1): viz-cluster1 ISA SET [InSet(SpecialHW, "11x5 tiled-display")]; Count(viz-cluster1)==25;
(2): viz-cluster2 ISA SET [InSet(SpecialHW, "4x2 tiled-display")]; Count(viz-cluster2)==8;
(3): str-cluster1 ISA SET [InSet(DataSet, "SoCalSAFS00-40")]; Count(str-cluster1) == 13;
(4): str-cluster2 ISA SET [InSet(DataSet, "SoCalSAFS41-80")]; Count(str-cluster2) == 7;
(5): str-cluster3 ISA SET [InSet(DataSet, "SoCalSAFS81-99")]; Count(str-cluster3) == 5;
(6): lambda1 ISA CONN (<viz-cluster1>, <str-cluster1>) [type="lambda"; bandwidth >= 10000];
(7): lambda2 ISA CONN (<viz-cluster1>, <str-cluster2>) [type="lambda"; bandwidth >= 6000];
(8): lambda3 ISA CONN (<viz-cluster1>, <str-cluster3>) [type="lambda"; bandwidth >= 4000]
```

Figure 7-4: A DVC-ISL specification for the collaborative visualization application

We started by evaluating the DVC resource selection and allocation performance. We used the DVC-ISL resource specification shown in Figure 7-4 and presented it to the DVC resource planning service (DVC-RCP). On the first run, the service returned with the physical resource configuration that includes 13 storage nodes at SOE/UCSD, 7 storage nodes at UIC, 5 storage nodes at UvA, 25 visualization nodes at CalIT2 and 8 visualization nodes at SIO. These particular storage nodes were picked because the requested datasets were present on their disks; as well they all had the required network bandwidth to the visualization cluster at

CalIT2. Even though the four UCSD storage clusters (SOM, SOE, CSE and SDSC) had better network connectivity to CalIT2 than that of UvA and UIC, only one UCSD storage cluster was chosen. This is due to the network hardware constraint at UCSD, which allows only one cluster to connect to CalIT2 at a time (controlled by the OXC switch). The returned resource configuration also includes the network configuration to establish the optical circuit path between the SOE storage cluster and the CalIT2 visualization cluster. The connections between the visualization cluster and the storage clusters (UvA and UIC) were pre-configured and static. Therefore, no dynamic network setup was required.

Next, the selected resources were allocated to create a new DVC computing environment. The DVC prototype employs the Globus GRAM for remote resource allocation and uses PIN/PDC for dynamic network configuration. In total, we allocated 58 storage and visualization nodes and set up one optical circuit path between CalIT2 and SOE. The entire resource configuration took 5.495 seconds. Table 7-2 breaks down the time in each step. We see that 14.94%, 60.55% and 24.51% is spent on the resource discovery and selection, end resource allocation, and dynamic network setup, respectively. The high resource allocation time is attributed to that fact that many storage nodes at UvA and UIC remotely from CalIT2 were allocated.

Table 7-2: The resource selection and allocation performance of the studied collaborative visualization application

Operation	Time (sec)	Percentage
Resource discovery and selection	0.821	14.94 %
End resource allocation	3.327	60.55 %
Dynamic network configuration	1.347	24.51 %
Total	5.495	

7.4.2 DVC Environment Configuration

To make application management of communication in heterogeneous and dynamic networks simpler, the DVC environment provides a virtual namespace. Once allocated, the CalIT2 visualization and SOE storage nodes were assigned private IP addresses to utilize the dynamically configured optical circuit. As a result, each visualization node at CalIT2 had two IP addresses: one for communication from/to UvA/UIC and another for communication from/to SOE. To shield the application from managing these heterogeneous addresses, the DVC assigned virtual IP addresses (and logical hostnames) to individual nodes. When these virtual addresses were used for communication, the DVC mapped them to the appropriate physical resource addresses and directs the traffic to the proper destination. Another advantage of this approach is application portability. Specifically, the same application can be run independently on different sets of physical resources because the same set of virtual names can be assigned to them. As an example of this portability, during these experiments we could replace the SOE storage clusters with other UCSD storage clusters (SOM, SDSC or CSE) and easily run the application.

7.4.3 Collaborative and Remote Data Visualization

Next, we built the collaborative visualization application on the DVC environment. The application inputted the name of datasets (collections of scene files) and transferred them on-demand from the three storage clusters (UvA, UIC and SOE) to the visualization cluster at CalIT2. Here, we used the DVC unified communication interface, which allows us to switch to use different transport protocols, including TCP, UDT and GTP. In our experiment, each of the 25 visualization nodes received a dataset of size 1.4 GB. The data transfer performance was measured when different protocols were used. For example, Figure 7-5 illustrates the

trajectory of the aggregate data aggregation rate when we ran the application with GTP. The data illustrates that GTP achieved a peak transfer rate of 16.3 Gbps out of the 20 Gbps available bandwidth, or 81.5% utilization.

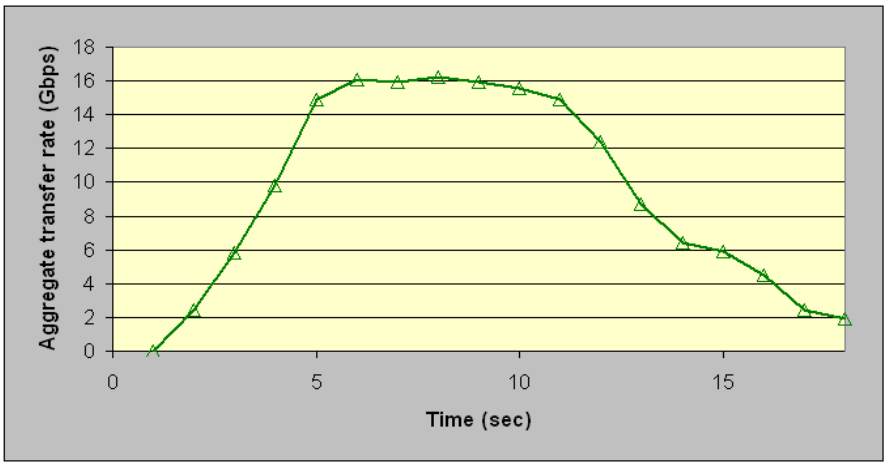


Figure 7-5: The trajectory of the aggregate transmission rate when running the collaborative visualization application with GTP

In the final step, the application visualized the transferred datasets with 'iView3d' on the visualization clusters at CalIT2 and SIO. It displayed multiple scene files in parallel on the tiled display panels, allowing the scientists at the two locations to simultaneously observe and analyze the correlation between different sets of data.

7.5 Summary

This chapter presented case studies for constructing collaborative visualization environments with the DVC prototype on the real, large-scale Lambda-Grid testbed. Our argument was that while such collaborative environments are feasible with emerging Grid and optical network infrastructures, many technical challenges remain. These challenges include: 1) on-demand discovery, selection and configuration of network and end resources; 2) development of applications on cross-domain, wide-area distributed environments; and 3)

management of heterogeneous resource names; and 4) use of novel exotic protocols to achieve high performance.

Our experiments demonstrated the key capabilities of the DVC that address these challenges for real application examples. These include: efficient resource discovery and selection, dynamic resource allocation and private network configuration, as well as a virtual resource namespace and unified communication interfaces for application flexibility and portability. Together, these capabilities enable collaborative visualization applications to be effectively and conveniently constructed in seconds and successfully exploit the novel communication capabilities of Lambda-Grids.

7.6 Acknowledgement

Chapter 7, in part, is published as "Collaborative Data Visualization for Earth Sciences with the OptIPuter" by Nut Taesombut, Xinran Wu, Andrew A. Chien, Atul Nayak, Bridget Smith, Debi Kilb, Thomas Im, Dane Samilo, Graham Kent and John Orcutt in Journal of Future Generation Computer Systems, Vol. 22(8), October 2006. The dissertation author was the primary researcher and co-author of this paper.

Chapter 8. Summary and Future Work

In this chapter, we summarize our research and results. Section 8.1 summarizes the coordinated resource management approach that enables integrated resource abstractions and combined resource selection. Section 8.2 presents the implications and impacts of our work. Finally, a number of possible directions for future work are outlined in Section 8.3.

8.1 Summary

Lambda-Grids provide intriguing opportunities for new computation, communication and collaboration capabilities at the cost of more heavy-weight, user-controlled resource management. Supporting easy and efficient development of high-performance applications in dynamic, heterogeneous and multi-institutional distributed resource environments is a critical challenge. Furthermore, utilizing network configurability presents unique challenges and adds the complexity of resource management and planning for networks to that for end compute or storage resources.

We propose the Distributed Virtual Computer (DVC), a novel integrated approach for managing network and end resources for high application performance and resource efficiency in Lambda-Grids. The DVC provides a simple service model – applications describe and acquire a dedicated set of communication and end resources, and subsequently make use of them as a private distributed presence to achieve quality of service, including high performance, synchronous collaboration and real-time. Key components of the DVC include: 1) a resource specification language describing combined application resource needs and exposing novel communication capabilities; 2) a resource planning service integrating end resource selection and network configuration optimization; 3) a resource binding service coordinating allocation of communication and end resources; and 4) a set of simplifying abstractions encapsulating a wealth of network and grid resource complexity. Altogether,

these services provide a simplified computing environment of the Lambda-Grid infrastructure with the complexity of use comparable to that of a private, local distributed system.

In the DVC model, a key challenge in enabling high application performance and efficient resource use is the selection of appropriate sets of resources for individual applications. The problem is formulated and several different resource selection strategies are explored. The key advantage of the DVC architecture is the integration of network configuration planning and end resource selection. We present two combined resource selection algorithms based on simulated annealing and top-down hierarchical selection. These are then evaluated via simulation across a range of realistic resource configurations and application models for Lambda-Grids. Our simulations demonstrate that the two combined selection approaches achieve good optimality for both application performance and network resource efficiency. Compared to traditional separate selection approaches, they produce better results for success selection rate, system throughput, network transmission, and setup cost. Further, the algorithm based on top-down hierarchical combined selection not only achieves good quality of results, but also scales well with both application request complexity and Lambda-Grid size.

For a large-scale Lambda-Grid system, the underlying network is typically partitioned into domains operated by different Internet Service Providers (ISPs). A key challenge is network information sharing that must not only enable efficient resource selection for grid applications, but also maintain competitive advantages of individual ISPs. This information sharing problem was studied for configurable optical networks and how the available information affects ISPs' and applications' ability to utilize communication resources was evaluated. Our simulation shows that 1) domain topology information is crucial for good resource efficiency, low network transmission and setup cost; 2) domain link state information contributes to better system throughput and utilization of lambdas; 3) when internal network

information cannot be shared, appropriate domain connectivity information helps improve system throughput and application communication latency; and 4) design of an ISP network has a strong impact on system lambda utilization and the utility of domain link state information. The first empirical data is presented here on the resource efficiencies of real ISPs and the ability of an application to utilize network resources with limited network information sharing.

To demonstrate the feasibility and advantages of the DVC idea, we develop the DVC system software prototype. The prototype builds on and leverages existing dynamic network provisioning tools and grid services, being innovative to enable applications to use novel communication capabilities of Lambda-Grids. To evaluate it based on real use with scientific applications, we employ the prototype to enable collaborative visualization environments for geosciences. We demonstrate the distributed scientific collaboration applications can be effectively and conveniently constructed on the OptIPuter's international-scale testbed and to successfully utilize the Lambda-Grid capabilities.

8.2 Implications

The main implication of our research is DVC coordinated resource management is a viable scheme enabling easy and efficient development of high-performance applications in Lambda-Grids. First, the presented DVC architecture demonstrates its feasibility and benefits in allowing applications to conveniently acquire and use resources in the distributed grid infrastructure and configurable networks. We have shown that scientific collaborative visualization environments can be constructed with the DVC implementation prototype in seconds across the international-scale Lambda-Grid testbed. Second, our simulation shows the combined resource selection schemes (enabled by the DVC architecture) can produce good results for success rate, resource quality, and network resource efficiency. Consequently, these

properties enable high application capabilities and system throughput. Third, we have demonstrated the combined approach based on hierarchical selection not only produces good results, but also scales well with the size of resource configurations past that of the currently deployed Lambda-Grids.

The second implication is deep understanding of network information sharing required for efficient resource selection for grid applications and effective traffic engineering for network service providers. Specifically, our simulation results demonstrate that providers' internal network information (including topology, link capacity and usage) is essential for better system throughput, network efficiency and application communication performance. These results suggest that collaboration between service providers can produce better overall network productivity as well as offer better network service to applications.

Our research enables a radical new type of distributed application paradigm that can exploit dedicated optical circuits to tightly couple geographically dispersed resources on-demand. With the DVC, applications express their communication and resource needs; and the DVC implementation configures network and end resources to support those needs. These resources are dedicated for use by applications and transparently managed for guaranteed, high performance and synchronous collaboration. This capability provides dramatic opportunities for new, innovative applications that involve large data objects and collections, large computations, high-performance visualization and wide-area collaboration. Examples of applications benefiting from such capabilities include:

- Distributed computational steering
- Collaborative and remote scientific data visualization
- Distributed scientific data distribution and sharing
- Distributed commercial content delivery, such as Akamai, BitTorrent and broadcast television

- Emerging distributed services involving large amount of content, indexing, etc., such as Google, Yahoo, etc.

Our research also provides a foundation for the use of coordinated network and resource management to support efficient development of high-performance applications in Lambda-Grids. Such coordinated management allows the integration of end resource selection and network configuration optimization. This improves both application capabilities and resource efficiencies. Existing dynamic network provisioning services and grid middleware manage communication and end resources separately, so such combined resource selection was impossible previously.

Finally, our work encourages network service providers to share their internal network information for better overall network productivity. Such information sharing improves their resource use efficiency and offers better network services to applications, thereby attracting more customers to the providers' networks. In fact, many Internet Service Providers (ISPs) (including AboveNet, NLR and Level3) publish this information today (completely or partially) and there are a range of research efforts [129-131] on inference techniques to effectively approximate ISP and Internet topologies. Therefore, other network providers are also encouraged to make this information available. Further, if complete internal network information cannot be shared, ISPs should provide approximate domain connectivity information which can improve network efficiency.

8.3 Future Work

Our research focused primarily on demonstrating the viability of the coordinated resource management approach in supporting efficient and easy development of high-performance applications for Lambda-Grids. Along this avenue, we studied the resource selection and network information sharing problems, evaluating different selection strategies

and the impacts of the available information on the quality of results. While we believe that we have made significant contributions in these areas, more advances can be made to improve our research. In this section, we identify several future research directions.

8.3.1 Deeper Simulation Studies

Further extensions can be made to improve the simulation studies of the resource selection and network information sharing problems. First, better application workloads can improve realism. While in our experiments we used synthetic workloads modeled after realistic scientific applications for Lambda-Grids, they may not perfectly represent real resource requests used in the infrastructures. Therefore, our studies can benefit from the real application workloads of emerging scientific distributed infrastructures such as BIRN and EarthScope. Second, broader multi-ISP network models can improve the fidelity of the study of the impact of network information models. The studied multi-ISP, Internet backbone network model was derived from ten large ISPs, and most pairs of these ISPs are directly peered. As a result, we observed little impact of interdomain network topology information on resource efficiency and applications. In fact, there are also a large number of small ISPs in the real Internet, and it would be interesting to see how these ISPs affect the utility of interdomain information. Third, broader application models can improve our understanding of the impact of the limited network information on applications and ISPs. Due to physical resource constraints, our experiments used the content distribution workload because each request is the simple selection of a replica server and a circuit path. Such simple requests enable us to have high enough numbers of requests in each workload trace to saturate the network and observe the effect of resource contention. Nevertheless, using more application models to evaluate the network information models would provide more insightful results. Forth, better resource selection algorithms can improve the evaluation of the impact of network information models.

The current simulation used only simple (greedy) algorithms to evaluate all models. While this method is used for fair comparison, there are opportunities to improve the attained results of each model by designing appropriate resource algorithms specific for it.

8.3.2 Automatic Generation of Application Resource Specification

In the DVC model, applications (or consequently application developers or users) explicitly describe and acquire the needed communication and end resources. A key challenge is understanding what applications require and prefer, and to create appropriate resource specifications to drive the resource selection and network configuration optimization. This is critical because the selection of suitable resources can make major performance difference for applications. However, constructing optimal resource specifications is difficult – it requires some experience with the underlying infrastructures and applications. As a result, high-level users (such as scientists) end up using sub-optimal specifications, thereby leading to limited application capabilities and/or inefficient resource use.

Therefore, a possible future research direction includes a framework to automatically generating optimal resource specifications for applications. This could be done by analyzing the structures of application implementation codes and/or the runtime behaviors of previous application runs. Experimenting with applications on different sets of resource configurations (either via simulation or real testbeds) would lead us to a prediction model for optimal application resource requirements.

8.3.3 Framework for Controlled Network Information Sharing

Our research identified information sharing as problematic for configurable optical networks and demonstrated that collaboration between ISPs can improve both overall network efficiency and productivity. A possible future research direction would include a framework

for controlled information sharing that maintains security and financial benefits of individual ISPs. To date, most ISPs (e.g., AT&T, Verizon, and Sprint) reply on the BGP protocol for exchanging interdomain route information and hide domain network information, or they (e.g., NLR and OptIPuter) provide full low-level domain information. These models have clear commercial limitations, and represent only two extreme points on the spectrum of possible design. Potential controlled information sharing models include some abstraction (approximation) of domain topologies, internal link connectivity and capacity. Simulation studies similar to our work can be used to evaluate the impact of these models on applications and ISPs.

8.3.4 More Application Experiments

In this dissertation, we developed the DVC system software prototype and used it to construct collaborative visualization environments for geosciences. Possibly, future work will evaluate it with broader types of scientific applications and with more complicated resource configurations. Potential applications include wide-area earthquake warning systems [127], remote scientific instrument control [128], scientific data distribution and sharing for biomedical research [5], etc. Additional Lambda-Grid testbeds include DRAGON [17], CHEETAH [18], Global Lambda Interchange Facility (GLIF) [19], CANARIE's CA*net 4 [20], and ISPs' dark fiber infrastructures. While the current prototype is robust and implements most key components of the DVC architectures, some challenges remain: 1) management of very large collections of resources and application components (i.e., several hundreds to thousands); 2) support for complex communication mechanisms such as multicast and distributed shared memory; and 3) support for hard QoS quality-of-service guarantee.

Appendix: Integrated Resource Specification Language Definition

The DVC integrated specification language (DVC-ISL) describes resource configurations (including network, computer, storage, visualization and other instruments) that host applications' computation and communication. This allows applications to share specific knowledge of their resource needs, and drive resource selection and network configuration optimization. Figure A-1 shows the full BNF description of the DVC-ISL. Table A-1 provides the syntax of its terminals (in the form of regular expression). Table A-2, A-3 and A-4 respectively list end-resource, connectivity and internal communication node attributes recognized by the DVC.

Table A-1: The syntax of terminals in the DVC-ISL BNF description

Token Name	Regular Expression	Example Value
Real	(-)?{0-9}*[\.]{0-9}+	`3.417, -0.0023`
Integer	(-)?{0-9}+	`12500, -39`
Boolean	(true | false)	`true, false`
Undefined	Undefined	`Undefined`
Error	Error	`Error`
String	\"[a-zA-Z0-9_\-\)\(\.]*\"	`"GNU/Linux"`
Identifier	[a-zA-Z_-][a-zA-Z0-9_-]*	`comp1, storage20`
Reference	\<[a-zA-Z0-9_\-\)\(\.]*\>	`<comp1>`

Table A-2: The list of end-resource attributes of the DVC-ISL

Attribute Name	Type	Description	Example Constraint
Hostname	String	Hostname/IP address	`Hostname == "192.168.82.2"`
CPUSpeed	Real	CPU clock speed (GHz)	`CPUSpeed > 2.4`
Platform	String	Computer platform name	`Platform == "x86_64"`
CPUModel	String	CPU model name	`CPUModel == "AMD Opteron(tm) Processor 246"`
CPUCache	Integer	CPU layer-2 cache size (MB)	`CPUCache >= 512`
CPUCount	Integer	Total number of CPUs	`CPUCount == 2`
MemoryTotal	Integer	Total memory size (MB)	`MemoryTotal >= 2048`
MemoryFree	Integer	Free memory size (MB)	`MemoryFree >= 1024`
DiskTotal	Integer	Total disk space (GB)	`DiskTotal > 200`
DiskFree	Integer	Free disk space (GB)	`DiskFree > 120`
OSName	String	Operating system name	`OSName == "Linux"`
SpecialHW	Set of string	List of attached hardware	`InSet(SpecialHW, "tiled-display")`
DataSet	Set of string	List of stored dataset	`InSet(DataSet, "seismic.scene")`

DVC-ISL-Spec ::= **Identifer** "**=**" *Specification*

Specification ::= "**[**"*StatementList* "**]**"

StatementList ::= *Statement* ["**;**" *Statement*]*

Statement ::= **Identifier** "**ISA**" "**[**" *ConstraintList* "**]**"
| **Identifier** "**ISA SET**" "**[**" *ConstraintList* "**]**"
| **Identifier** "**ISA CONN**" "**(**" *ReferenceList* "**)**" "**[**" *ConstraintList* "**]**"
| **Identifier** "**ISA CNODE**" "**[**" *ConstraintList* "**]**"
| *Predicate*
| *Constraint*

ConstraintList ::= *Constraint* ["**;**" *Constraint*]*

Constraint ::= **Identifer** "**=**" *LogicalExpr*
| **Identifer** "**=**" "**ENUM**" "[" *PrimitiveTypeList*"]"
| **Identifer** "**=**" "**DICTIONARY**" "[" *DicElementList* "]"
| **Identifer** "**=**" "**[**" *ExprSet* "**]**"
| **Identifer** "**=**" "**{**" *ExprList* "**}**"
| *LogicalExpr*

LogicalExpr ::= *RelationalExpr*
| *LogicalExpr* ("**&&**" | "||") *RelationalExpr*
| "**Required**" "**(**" *ExprSet* "**)**"

RelationalExpr ::= *ArithExpr*
| *RelationalExpr* (">" | "<" | ">=" | "<=" | "==" | "**!**=") *ArithExpr*

ArithExpr ::= *AggrOprExpr*
| *ArithExpr* ("+" | "**-**" | "/" | "**%**" | "**&**" | "|" | "^" | "<<" | ">>") *AggrOprExpr*

AggrOprExpr ::= *SetOprExpr*
| ("**Count**" | "**Min**" | "**Max**" | "**Avg**") "**(**" *PosfixExpr* "**)**"

SetOprExpr ::= *UnaryExpr*
| ("**InSet**" | "**Set_Intersection**" | "**Set_Union**" | "**Set_Difference**" | "**Set_S_Difference**") "**(**" *PostfixExpr* "**,**" *LogicalExpor* "**)**"

UnaryExpr ::= *PostfixExpr*
| ("+" | "**-**" | "**!**") *UnaryExpr*

PostfixExpr ::= *TypeExpr*
| **Identifer** ["**.**" **Identifer**] ["**[**" *PostfixExpr* "**]**"]

TypeExpr ::= *PrimitiveType*
| **Identifer** | "**(**" *LogicalExpr* "**)**"

ExprList ::= *LogicalExpr* ("**,**" *LogicalExpr*)*

ExprSet ::= *LogicalExpr* ("**,**" *LogicalExpr*)*

DicElementList ::= *DicElement* ("**,**" *DicElement*)*

DicElement ::= "**{**" **String** "**,**" *PrimitiveType* "**}**"

Predicate ::= ("**Maximize**" | "**Minimize**") "**(**" *ArithExpr* "**)**"
| ("**Forall**"| "**Forany**") **Identifer** "**in**" **Identifer**

ReferenceList ::= *Reference* ["**,**" *Reference*]*

PrimitiveTypeList ::= *PrimitiveType* ("**,**" *PrimitveType*)*

PrimitveType ::= **Real** | **Integer** | **String** | **Boolean** | **Reference** | **Undefined** | **Error**

Figure A-1: The full BNF description of the DVC integrated specification language

Table A-3: The list of network connectivity attributes of the DVC-ISL

Attribute Names	Type	Description	Example Constraint
type	String	Connectivity type (either "internet", "lambda", or "intra-cluster)	`type = "lambda"`
Bandwidth	Real	Bandwidth of individual connections (Mbps)	`Bandwidth > 2000`
Latency	Real	Latency (ms)	`Latency < 20`

Table A-4: The list of internal communication node attributes of the DVC-ISL

Attribute Name	Type	Description	Example Constraint
Hostname	String	Hostname/IP address	`Hostname == "172.31.20.1"`
exchange	-	Allow aggregation of traffic	`Required(exchange)`
opt-multicast	-	Allow duplication of an optical signal as-is to enable multicast (one-way)	`Required(opt-multicast)`

References

[1] V. Astakhov, A. Gupta, S. Santini, and J. S. Grethe, "Data Integration in the Biomedical Informatics Research Network (BIRN)," in *Proceedings of the 2nd International Workshop on Data Integration in Life Sciences (DILS'05)*, July 2005.

[2] J. M. Brooke, P. V. Coveney, J. Harting, S. Jha, S. M. Pickles, R. L. Pinning and A. R. Porter, "Computational Steering in RealityGrid", in *Proceedings of the UK e-Science All Hands Meeting*, September 2003.

[3] T. A. DeFanti, J. Leigh, et al., "Teleimmersion and Visualization with the OptIPuter," in *Proceedings of the 12th International Conference on Artificial Reality and Telexistence (ICAT'2002)*, December 2002.

[4] EarthScope. http://www.earthscope.org

[5] BIRN – Biomedical Informatics Research Network. http://www.nbirn.net

[6] ORION – Ocean Research Interactive Observatory Networks. http://www.orionprogram.org

[7] J. Bunn and H. Newman, "Data-Intensive Grids for High-Energy Physics," in *Grid Computing: Making the Global Infrastructure a Reality*," F. Berman, G. Fox, and T. Hey, Eds. John Wiley & Sons, Inc., New York, 2003.

[8] AstroGrid. http://www2.astrogrid.org

[9] I. Foster and C. Kesselman, editors, "The Grid: Blueprint for a New Computing Infrastructure," Morgan Kaufmann, 1999.

[10] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of Grid: Enabling Scalable Virtual Organizations," *International Journal of Supercomputer Applications*, 15(3), 2001.

[11] TeraGrid. http://www.teragrid.org

[12] iVDGL – International Virtual Data Grid Laboratory. http://www.ivdgl.org

[13] GriPhyN – Grid Physics Network. http://www.griphyn.org

[14] The DataGrid Project. http://eu-datagrid.web.cern.ch/eu-datagrid

[15] L. L. Smarr, A. A. Chien, T. A. DeFanti, J. Leigh, and P. M. Papadopoulos, "The OptIPuter," *Communications of the ACM*, Vol. 46(11), November 2003. http://www.optiputer.net

[16] National LambdaRail. http://www.nlr.net

[17] T. Lehman, J. Sobieski, and B. Jabbari, "DRAGON: A Framework for Service Provisioning in Heterogeneous Grid Networks," *IEEE Communications Magazine*, Vol. 44(3), March 2006. http://dragon.east.isi.edu

[18] X. Zheng, M. Veeraraghavan, N. S. V. Rao, Q. Wu, and M. Zhu, "CHEETAH: Circuit-Switched High-Speed End-to-End Transport Architecture Testbed," *IEEE Communication Magazine*, Vol. 43(8), August 2005.

[19] Global Lambda Integrated Facility. http://www.glif.is

[20] CANARIE's CAnet 4. http://www.canarie.ca/canet4

[21] NetherLight. http://www.netherlight.net

[22] T. DeFanti, C. de Laat, J. Mambretti, K. Neggers, and B. St. Arnaud, "TransLight: A Global-Scale LambdaGrid for e-Science," *Communications of the ACM*, Vol. 26(11), November 2003.

[23] R. Wu and A. A. Chien, "GTP: Group Transport Protocol for Lambda-Grids," in *Proceedings of the 4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'2004)*, April 2004.

[24] Y. Gu and R. L. Grossman, "UDT: UDP-Based Data Transfer for High-Speed Wide Area Networks," *Journal of Computer Networks*, Vol. 51(7), May 2007.

[25] E. Weigle and A. A. Chien, "The Composite Endpoint Protocol (CEP): Scalable Endpoints for Terabit Flows," in *Proceedings of the IEEE Conference on Cluster Computing and the Grid (CCGrid'2005)*, April 2005.

[26] I. Foster, "Globus Toolkit 4: Software for Service-Oriented Systems," in *Proceedings of the IFIP International Conference on Network and Parallel Computing*, Springer-Verlag, LNCS 3779, 2006. http://www.globus.org

[27] Y. Kee, D. Logotheis, R. Huang, H. Casanova, and A. A. Chien, in *Proceedings of the IEEE Conference of Cluster Computing and the Grid (CCGird'2005)*, April 2005.

[28] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids," in *Proceedings of the 10th IEEE Symposium on High Performance Distributed Computing*, August 2001.

[29] Lightweight Middleware for Grid Computing (gLite). http://www.glite.org

[30] G. Allen, K. Davis, et al., "Enabling Applications on the Grid – A GridLab Overview," *International Journal of High Performance Computing Applications*, August 2003.

[31] O. Yu, A. Li, Y. Cao, L. Yin, M. Liao, and H. Xu, "Multi-Domain Lambda Grid Data Portal for Collaborative Grid Applications," *Journal of Future Generation Computer Systems*, Vol. 22(8), October, 2006.

[32] J. Wu, S. Campbell, J. M. Savoie, H. Zhang, G. Bochmann, and B. St.Arnaud, "User-Managed End-to-End Lightpath Provisioning over CA*net4," in *Proceedings of the National Fiber Optic Engineers Conference (NFOEC)*, September 2003.

[33] Y. Kee, H. Casanova, and A. A. Chien, "Realistic Modeling and Synthesis of Resources for Computational Grids," in *Proceedings of the ACM Conference on High Performance Computing and Networking (SC'04)*, November 2004.

[34] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, "A Security Architecture for Computational Grids," in *Proceedings of the 5th ACM Conference on Computer and Communication Security Conference*, 1998.

[35] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid Information Services for Distributed Resource Sharing," in *Proceedings of the 10th IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)*, August 2001.

[36] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu and I. Foster, "The Globus Striped GridFTP Framework and Server," in *Proceedings of the ACM Conference on High Performance Computing and Networking (SC'2005)*, November 2004.

[37] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke, "A Resource Management Architecture for Metacomputing Systems," in *Proceedings of the IPPS/SPDP Workshop on Job Scheduling Strategies for Parallel Proceedings*, 1998.

[38] The WS-Resource Framework. http://www.globus.org/wsrf/

[39] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy, "A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation," in *Proceedings of the International Workshop on Quality of Service*, 1999.

[40] T. Tannenbaum, D. Wright, K. Miller, and M. Livny, "Condor – A Distributed Job Scheduler," in Thomas Sterling, editor, *Beowulf Cluster Computing with Linux*, the MIT Press, 2002.

[41] LHC Computing Grid Project. http://lcg.web.cern.ch/LCG/

[42] P. Badino and P. Kunszt, "gLite I/O User's Guide v.1," EGEE Technical Report 570771, March 2005.

[43] GridLab Project: W-9 Resource Management.
http://www.gridlab.org/WorkPackages/wp-9/

[44] GridLab Project: W-1 Grid Application Toolkit.
http://www.gridlab.org/WorkPackages/wp-1/

[45] J. Leigh, L. Renambot, T. A. DeFanti, M. Brown, E. He, N. Krishnaprasad, J. M. A. Meerasa, A. Nayak, K. Park, R. Singh, S. Venkataraman, and C. Zhang, "An Experimental OptIPuter Architecture for Data-Intensive Collaborative Visualization," in *Proceedings of the 3rd Workshop on Advanced Collaborative Environments (WACE'03)*, June 2003.

[46] H. Liu, D. Pendarakis, N. Komaee, and D. Saha, "GMPLS-Based Control Plane for Optical Networks: Early Implementation Experience," in *Proceedings of the International Symposium and Exhibit on the Convergence of Information Technology and Communications*, August 2002.

[47] A. P. Mudambi, X. Zheng, and M. Veeraraghavan, "A Transport Protocol for Dedicated End-to-End Circuits," in *Proceedings of the IEEE International Conference on Communications (ICC'06)*, June 2006.

[48] M. Veeraraghavan, H. Lee, E. K. P. Chong and H. Li, "A Varying Bandwidth List Scheduling Heuristic for File Transfers," in *Proceedings of the IEEE International Conference on Communications (ICC'04)*, June 2004.

[49] J. Vollbrecht, P. Calhoun, et al., "AAA Authorization Framework," RFC 2904, August 2000.

[50] DRAGON Project: Network Aware Resource Broker (NARB). http://dragon.east.isi.edu/twiki/bin/view/Main/NARB

[51] Photonic Domain Controller. http://www.evl.uic.edu/cavern/rg/20031003_he

[52] R. Raman, M. Livny, and M. Solomon, "Matchmaking Distributed Resource Management for High Throughput Computing," in *Proceedings of the 7th IEEE Symposium on High Performance Distributed Computing (HPDC-7)*, July 1998.

[53] R. Raman, M. Livny, and M. Solomon, "Policy Driven Heterogeneous Resource Co-Allocation with Gangmatching," in *Proceedings of the 12th IEEE Symposium on High Performance Distributed Computing (HPDC-12)*, June 2003.

[54] C. Liu and I. Foster, "A Constraint Language Approach to Matchmaking," in *Proceedings of the 14th Workshop on Research Issues on Data Engineering: Web Services for E-Commerce and E-Government Applications*, March 2004.

[55] K. Marriott and J. S. Peter, *Programming with Constraints: An Introduction*, The MIT Press, Cambridge, Massachusetts, 1998.

[56] H. Tangmunarunkit, S. Decker, and C. Kesselman, "Ontology-based Resource Matching in the Grid – The Grid Meets the Semantic Web," in *Proceedings of the International Semantic Web Conference (ISWC'2003)*, October 2003.

[57] S. Decker and M. Sintek, "Triple – A Query, Inference, and Transformation Language for the Semantic Web," in *Proceedings of the 13th International Semantic Web Conference (ISWC'2002)*, June 2002.

[58] P. Dinda and D. Lu, "Nondeterministic Queries in a Relational Grid Information Service," in *Proceedings of the ACM Conference on High Performance Computing and Networking (SC'03)*, November 2003.

[59] S. Fisher, "Relational Model for Information and Monitoring," Technical Report GWD-GP-7-1, 2001.

[60] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat, "Design and Implementation Tradeoffs for Wide-Area Resource Discovery," in *Proceedings of the 14th IEEE Symposium on High Performance Distributed Computing*, July 2005.

[61] A. Huang and P. Steenkiste, "Building Self-Configuring Services using Service-Specific Knowledge," in *Proceedings of the 13th IEEE International Symposium of High-Performance Distributed Computing (HPDC-13)*, June 2003.

[62] R. Ricci, C. Alfeld, and J. Lepreau, "A Solver for the Network Testbed Mapping Problem," *ACM SIGCOMM Computer Communication Review*, Vol. 33(2), April 2003.

[63] J. Londono and A. Bestavros, "NetEMBED: A Network Resource Mapping Service for Distributed Applications," Boston University Technical Report (BUCS-2006-032), December 2006.

[64] The Globus Resource Specification Language (RSL) v1.0.
http://www.globus.org/toolkit/docs/2.4/gram/rsl_spec1.html

[65] The ClassAd Language Reference Manual v2.4.
http://www.cs.wisc.edu/condor/classad/refman.pdf

[66] C. Liu and I. Foster, "A Constraint Language Approach to Grid Resource Selection," University of Chicago Technical Report TR-2003-07, 2003.

[67] D. Oppenheimer, J. Albrecht, D. Patternson, and A. Vahdat, "Scalable Wide-Area Resource Discovery," UC Berkeley Technical Report UCB//CSD-04-1334, July 2004.

[68] A. A. Chien, H. Casanova, Y. Kee, and R. Huang, "The Virtual Grid Description Language: vgDL," UCSD Technical Report CS2005-0817, 2005.

[69] E. He, J. Leigh, O. Yu, and T. A. DeFanti, "Reliable Blast UDP: Predictable High Performance Bulk Data Transfer," in *Proceedings of the IEEE Cluster Computing (Cluster'2002)*, September 2002.

[70] V. Vishwanath, J. Leigh, E. He, M. D. Brown, L. Long, L. Renambot, A. Verlo, X. Wang, and T. A. DeFanti, "Wide-Area Experiments with LambdaStream over Dedicated High-Bandwidth Networks," in *Proceedings of the 25th Conference on Computer Communications (INFOCOMM'2006)*, April 2006.

[71] Z. Huang, L. Gu, B. Du, and C He, "Grid Resource Specification Language Based on XML and Its Usage in Resource Registry Meta-Service," in *Proceedings of the 2004 IEEE International Conference on Services Computing (SCC'04)*, September 2004.

[72] L. Pearlman, V. Welch, I. Foster, and C. Keselman, "A Community Authorization Service for Group Collaboration," in *Proceedings of the 3rd IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*, June 2002.

[73] Y. Kee, K Yocum, A. A. Chien, H. Casanova, "Improving Grid Resource Allocation via Integrated Selection and Binding," in *Proceedings of ACM/IEEE International Conference on High Performance Computing and Communication (SC'06)*, November 2006.

[74] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," Science, Vol. 200(4598), May 1983.

[75] X. Zhang and J. Y. Wei, "Constrained Multicast Routing in WDM Networks with Sparse Light Splitting," *Journal of Lightwave Technology*, Vol. 18(12), December 2000.

[76] K. Marriott and J. S. Peter, "Programming with Constraints: An Introduction," the MIT Press, Cambridge, 1998.

[77] MySQL. http://www.mysql.com

[78] G. Karypis and V. Kumar, "A Fast and Highly Quality Multilevel Scheme for Partitioning Irregular Graphs," *SIAM Journal on Scientific Computing*, Vol. 20(1), December 1998.

[79] Verizon Business: Global Network Map.
http://www.verizonbusiness.com/us/about/network

[80] Portable Batch Scheduler. http://www.openpbs.org

[81] Sun Grid Engine. http://gridengine.sunsource.net

[82] D. B. Hoang, T. Lavian, S. Figueira, J. Mambretti, I. Monga, S. Naikasatam, H. Cohen, D. Cutrell and F. Travostino, "DWDM-RAM: An Architecture for Data Intensive Service Enabled by Network Generation Dynamic Optical Networks," in *Proceedings of the IEEE Workshop on High-Performance Global Grid Networks*, November 2004.

[83] BigBangwidth Lightpath Accelerator System version 2.5.2, User and Installation Guide.

[84] AT&T Global Network Map. http://www.corp.att.com/globalnetworking

[85] BT Global Services: Network Maps. http://www.bt.net/info/maps.shtml

[86] Cogent Communication: Network Map. http://www.cogentco.com/htdocs/map.php

[87] Global Crossing Interactive Network Map.
http://www.globalcrossing.com/network/network_interactive_map.aspx

[88] The Level3 Network.
http://www.level3.com/images/global_map/Level_3_Network_map.pdf

[89] NTT Communications: Global IP Network.
http://www.ntt.net/english/about/network_map.html

[90] Qwest Network Maps. http://www.qwest.com/about/qwest/network/index.html

[91] Sprint Global IP Network. http://www.sprintworldwide.com/english/maps

[92] Time Warner Telecom IP Network Map.
http://www.twtelecom.com/about_us/lg_ip_map.html

[93] Rocketfuel: An ISP Topology Mapping Engine.
http://www.cs.washington.edu/research/networking/rocketfuel

[94] N. Spring, R. Mahajan, and Wetherall, "Measuring ISP Topologies with Rocketfeul," in *Proceedings of the ACM Conference of the Special Interest Group on Data Communication (SIGCOMM'02)*, August 2002.

[95] CAIDA's Inferred AS Relationships Dataset. http://as-rank.caida.org/data

[96] X. Dimitropolos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, K. C. Claffy, and G. Riley, "AS Relationships: Inference and Validation," *ACM SIGCOMM Computer Communication Review (CCR)*, Vol. 37(1), January 2007.

[97] W. F. Kern and J. R. Bland, Solid Mensuration with Proofs, Second Edition, John Wiley & Sons, 1954.

[98] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An Approach to Universal Topology Generator," in *Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS'01)*, August 2001.

[99] D. Churchill, S. Padina, and R. P. Bording, "Seismic Tomography as a High Performance Application," in *Proceedings of the 20th International Symposium on High-Performance Computing in an Advanced Collaborative Environment (HPCS'06)*, May 2006.

[100] P. Wilmott, "Derivatives: The Theory and Practice of Financial Engineering", John Wiley & Sons, 1998.

[101] J. Taylor, M. Dvorak, and S. Mickelson, "Developing Grid Based Infrastructure for Climate Modeling," in *Proceedings of the International Conference on Computational Science (ICCS'02)*, April 2002.

[102] B. M. E. Moret, D. A. Bader, and T. Warnow, "High-Performance Algorithm Engineering for Computational Phylogenetics," *Journal of Supercomputing*, Vol. 22(1), May 2002.

[103] S. Park, J. Lee, and S. Hariri, "A Multithreaded Message-Passing System for High Performance Distributed Computing Applications," in *Proceedings of the 18th International Conference on Distributed Computing Systems*, May 1998.

[104] A. Bilgin and M. W. Marcellin, "JPEG2000 for Digital Cinema," in *Proceedings of the 2006 IEEE International Symposium on Circuits and Systems (ISCAS'2006)*, May 2006.

[105] J. Kangasharju, J. Roberts, and K. W. Ross, "Object Replication Strategies in Content Distribution Networks," *Computer Communications*, Vol. 25(4), April 2002.

[106] G. M. Bernstein, V. Sharma, and L. Ong, "Interdomain Optical Routing," *Journal of Optical Networking*, Vol. 1(2), February 2002.

[107] J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher, Internet Denial of Service: Attack and Defense Mechnisms, Prentice Hall PTR, 2005.

[108] Z. L. Zhang, Z. Duan, and Y. T. Hou, "On Scalable Network Resource Management Using Bandwidth Brokers," in *Proceedings of the 8th Network Operations and Management Symposium (NOMS'2002)*, April 2002.

[109] Y. Rekhter and P. Gross, "Application of the Border Gateway Protocol in the Internet," RFC 1772, T. J. Watson Research Center, IBM Corporation, MCI March 1995.

[110] The ATM Forum Technical Committee, "Private Network-Network Specification Interface v.1 (PNNII 1.0)," af-pnni-0055.000, March 1996.

[111] AboveNet IP and Fiber Maps. http://www.above.net/products/maps2/index.html

[112] V. N. Padmanabhan and L. Subramanian, "An Investigation of Geographic Mapping Techniques for Internet Hosts," in *Proceedings of the ACM Conference of the Special Interest Group on Data Communication (SIGCOMM'01)*, August 2001.

[114] N. Taft, S. Bhattacharyya, J. Jetcheva, and C. Diot, "Understanding Traffic Dynamics at a Backbone POP," in *Proceedings of the SPIE ITCom Workshop on Scalability and Traffic Control IP Networks*, July 2001.

[115] N. Spring, R. Mahajan, and T. Anderson, "Quantifying the Causes of Path Inflation," in *Proceedings of the ACM Conference of the Special Interest Group on Data Communication (SIGCOMM'03)*, August 2003.

[116] X. Dimitropoulos, D. Krioukov, G. Riley, and K. C. Claffy, "Revealing the Autonomous System Taxonomy: The Machine Learning Approach," in *Proceedings of the 7th Passive and Active Measurements Workshop (PAM'06)*, April 2006.

[117] H. B. Newman, M. H. Ellisman, J. A. Orcutt, "Data-Intensive E-Science Frontier Research," *Communications of the ACM*, Vol. 46(11), November 2003.

[118] B. Jeong, L. Renambot, R. Jagodic, R. Singh, J. Aguilera, A. Johnson, and J. Leigh, "High-Performance Dynamic Graphics Streaming for Scalable Adaptive Graphics Environment," in *Proceedings of the ACM/IEEE Conference on High Performance Networking and Computing (SC'06)*, November 2006.

[119] A. Hutanu, G. Allen, et al., "Distributed and Collaborative Visualization of Large Data Sets using High-Speed Networks," in *Journal of Future Generation Computer Systems*, Vol. 22(8), October 2006.

[120] J. Leigh, L. Renambot, et al., "The Global Lambda Visualization Facility: An International Ultra-high-definition Wide-area Visualization, in *Journal of Future Generation Computer Systems*, Vol. 22(8), October 2006.

[121] J. Jo, W. Hong, et al., "Interactive 3D HD Video Transport for E-Science Collaboration over UCLP-enabled GLORIAD Lightpath, in *Journal of Future Generation Computer Systems*, Vol. 22(8), October 2006.

[122] IVS 3D – Fledermaus Professional.
http://www.ivs3d.com/products/fledermaus/fledermaus_pro.html

[123] IVS 3D – iView3D. http://www.ivs3d.com/products/iview3d/

[124] N. Freed, "Behavior of and Requirements for Internet Firewalls," RFC 2979, October 2000.

[125] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear, "Address Allocation for Private Internets," RFC 1918, February 1996.

[126] P. Grosso, P. de Boer, and L. Winkler, "The Network Infrastructure at iGrid2005: Lambda Networking in Action," in *Journal of Future Generation Computer Systems*, Vol. 22(8), October 2006.

[127] K. H. Kim, "Wide-Area Real-Time Computing in a Tightly Managed Optical Grid – An OptIPuter Vision," in *Proceedings of the 18th IEEE International Conference on Advanced Information Networking and Applications*, March 2004.

[128] T. E. Molina, G. Yang, A. W. Lin, S. T. Peltier, and M. H. Ellisman, "A Generalized Service-Oriented Architecture for Remote Control of Scientific Imaging Instruments," in *Proceedings of the 1st IEEE International Conference of e-Science and Computing*, December 2005.

[129] B. Wong, A. Slivkins, E. G. Sirer, "Meridian: A Lightweight Network Location Service without Virtual Coordinates," in *Proceedings of the ACM Conference of the Special Interest Group on Data Communication (SIGCOMM'05)*, August 2005

[130] M. den Burger, T. Kielmann, H. E. Bal, "TopoMon: A Monitoring Tool for Grid Network Topology," in *Proceedings of the 2002 International Conference on Computational Science*, April 2002.

[131] H. V. Madhyastha, T. Isdal, et al., "iPlane: An Information Plane for Distributed Services," in *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*, November 2006.